Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Leon Košak

Primerjava učinkovitosti izvajanja programske opreme na večjedrnih sistemih in splošnonamenskih grafičnih procesorjih

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Leon Košak, z vpisno številko **63060136**, sem avtor diplomskega dela z naslovom:

Primerjava učinkovitosti izvajanja programske opreme na večjedrnih sistemih in splošnonamenskih grafičnih procesorjih

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 1. julija 2014 Podpis avtorja:

Hvala družini za podporo v času študija in pisanja diplomskega dela ter asistentu Bojanu Klemencu, ki me je usmerjal in prispeval res velik delež svojega časa in tuda. Zahvaljujem se g. Andreju Ostermanu, ki mi je posredoval izvorno kodo aplikacije r.cuda.los, da sem lahko opravil primerjavo zmogljivosti procesoranja na realni aplikaciji. Zahvala pa gre tudi prof. dr. Patriciu Buliću, ki je sprejel mojo prošnjo za mentorstvo in me vodil do zaključka študija.

Diploma je posvečena staršema Zvonki in Franciju.

Kazalo

Povzetek

Količina podatkov vnešenih v različne računalniške sisteme narašča zelo hitro, zato je pomembno, da jih znamo pravilno interpretirati in jih organizirati. Z naraščanjem količine podatkov postaja potreba po hitrejšem procesiranju vedno večja, saj je izračunana končna informacija nepomembna, če je do ciljnega akterja v računalniškem sistemu prišla prepozno.

Grafični procesorji so se do časa pisanja diplomskega dela razvili do te mere, da ne omogočajo le specializiranega dela z računalniško grafiko, ampak tudi splošnonamensko procesiranje podatkov. Tudi splošnonamenske procesorje že nekaj let izdelujejo v večjedrnih izvedenkah, saj je ekonomsko ceneje izdelovati čipe z več jedri, kot z enim samim hitrim.

Diplomsko delo obravnava pregled trenutnega stanja področja paralelnega izvajanja računanskih operacij in ocenjevanje prihranjene električne energije za enako opravljeno delo s procesiranjem na grafični kartici. Test obsega testiranje dveh različnih aplikacij na dveh operacijskih sistemih. Aplikacija potujočega trgovskega potnika je predstavljena kot primer za prikaz gole procesne moči procesorjev, modula za geoinformacijski sistem pa kot primer resnično koristne aplikacije. Zaključek obsega še predloge izboljšav na programski ravni za še boljšo izkoriščanje procesne moči CPE in GPE.

Ključne besede: splošnonamensko računanje na grafičnih procesorjih (GPGPU), OpenCL, CUDA, podatkovne baze, paralelno izvajanje programske kode, GPE, večjedrna CPE

Abstract

The amount of data recorded in various computer systems is increasing rapidly, therefore it is important to know how to interpret and organize them correctly. With the increase of the quantity of data, the need for faster processing is increasing as well, since calculated final information is irrelevant if it is delivered to the final actor in the computer system too late.

By the time of writing this diploma thesis, graphics processing units have developed to the point where they enable not only specialized computer graphics manipulation but also general-purpose data processing. For some years now, general-purpose processors have been made in multicore models, since it is economically cheaper to make chips with more cores than to make them with one fast core.

The thesis presents an overview of the current situation in the field of parallel computing operations execution, and an estimate of electric energy savings at the same amount of work done, by processing on graphics card. The test includes tests of two different applications on two operating systems. The application of a travelling sales representative is presented as an example of a display of pure processing power of processors, and the two modules for geoinformation system as an example of a genuinely useful application. The conclusion of the thesis includes suggestions for improvements at program level for yet better exploitation of processing powers of CPU and GPU.

Key words: General purpose computing on graphics processing units (GPGPU), OpenCL, CUDA, Databases, parallel code execution, GPU multicore CPU

Slovar

Bruteforce Angleški izraz za način reševanja problemov, ki se omejuje na golo računsko zmogljivost procesiranje.

CPU Centralna procesna enota je osrednji procesor računalniškega sistema. V angleščini se zanj uporablja kratica CPU (Central Processing Unit), v slovenščini pa CPE (Centralna Procesna Enota).

HPC Veja v računalništvu, ki se ukvarja z visikozmogljivim procesiranjem med katere se po letu 2007 prišteva tudi splošnonamensko procesiranje na grafičnih karticah. Kratica HPC izhaja iz angleščega jezika in pomeni High-Performance Computing.

GPGPU Angleška kratica za General Purpose GPU, kar se v slovenski jezik prevaja kot grafični procesor za splošnonamensko procesiranje.

GPU Angleška kratica za grafični procesor (Graphics Processing Unit). V slovenščini obstaja analogno poimenovanje GPE (Grafična Procesna Enota).

Opravilo Največkrat se to besedo v računalniškem izrazoslovju v angleščini poimenuje kot task. Opravilo je neka smiselna celota, ki pa je lahko delček nekega večjega opravila.

HyperThreading Tehnologija podjetja Intel, ki ine fizično procesorsko jedro predstavi kot dva. Omenjena tehnologija pride do izraza predvsem v aplikacijah, ki zmorejo učinkovito izkoriščati več procesorskih jeder. Namen tehnologije je, da se že na najnižjem nivoju optimizira efektivnost procesiranja procesorskega jedra.

Pomnilnik s samodejnim popravljanjem napak Pomnilniki v strežniških sistemih morajo dostikrat zadostiti pogoju, da zaznajo in samo-

dejno popravijo določeno število napak, ki so posledica kozmičnih žarkov. V angleščini se take pomnilnike poimenuje s kratico ECC, kar pomeni Error Checking & Corrention.

Predpomnilnik Predpomnilnik je običajno hiter a prostorsko omejen pomnilnik, ki se arhitekturno nahaja bližje procesorju z namenom, da mu hitreje zagotavlja podatke, kot če bi jih bral naposredno iz največjega (in posledično najpočasnejšega) pomnilnika. Predpomnilnik se v angleščino prevaja kot cache.

Programski vmesnik Vmesnik, ki abstrahira in skrije določene tehnične implementacije nasptornega sistema z namenom poenostavljene interakcije z njim. V angleščini se ponavadi to poimenuje s kratico API (Application Programming Interface).

Renderiranje Izris oz upodobitev nekega grafičnega modela v rasterski izgled (video ali statično sliko) glede na osvetlitev in materiale, ki so definirani v omenjenem modelu. Tako upodabljanje je s stališča procesne moči zelo intenzivno.

Streaming Multiprocessor Enota večih procesorskih jeder pri grafičnem čipu, ki so združeni v eno enoto. Vsak načrtovalec grafičnih čipov ta termin poimenuje malenkost drugače, vendar gra v praksi za zelo podobno enoto grafičnega procesorja.

Ščepec Programska koda, ki se v več programskih nitih paralelno izvaja na grafičnem čipu. V angleščini je ščepec poimenovan z izrazom kernel.

TDP Kratica, ki v angleščini pomeni Thermal Design Power se v slovenščino prevaja kot toplotni pečat. V praksi to pomeni koliko energije največ lahko porablja nek čip v časovni enoti. Toplotni pečat se meri v merski enoti Watt.

Poglavje 1

Uvod

Že od pojava elektronskih računalniških sistemov potreba po hitrejšem procesiranju podatkov narašča zelo hitro. Količina podatkov v večini informacijskih sistemov se podvoji v manj kot 24 mesecih [1], zato morajo tudi procesne enote, ki procesirajo vse te podatke v uporabniku koristno informacijo, povečevati svojo zmogljivost vsaj tako hitro, kot narašča količina podatkov.

Na hitro rast količine podatkov v računalniških sistemih najbolj vpliva povečevanje enot za shranjevanje podatkov. Tem se je na dve leti kapaciteta podvajala, cena na enoto pa se je v enakem obdobju približno razpolovila. [2]

Najbolj znani predstavniki naprav za shranjevanje so magnetni trdi diski, ki pa jih v zadnjem obdobju vse bolj izpodrivajo naprave s pomnilniškimi celicami, t.i. *bliskovnimi pomnilniki*. Slednji nimajo gibljivih sestavnih delov, kot jih imajo klasični magnetni trdi diski, kar precej poveča zanesljivost in predvsem hitrost delovanja.

V drugem poglavju sta na kratko predstavljeni arhitekturi večjedrnega osrednjega procesorja (CPE) in grafičnega procesorja (GPE). Poznavanje različnih zgradb procesorjev je pomembno, saj ponazarja, zakaj je smiselno razvijati namenske procesorje za različne vrste problemov.

Tretje poglavje predstavlja začetke in razvoj paralelnega procesiranja podatkov na osrednjem procesorju. Računalniškega sistema si skorajda ni mo-

goče predstavljati brez splošnonamenskega osrednjega procesorja in dokler so bili na trgu prisotni le-ti, se je hitrost procesiranja morala povečevati zelo hitro. S tem so snovalci procesorske arhitekture hitro prišli na idejo o možnosti paralelnega procesiranja, ki pa prinaša manjše in večje težave. Teh se dotaknemo v enem od podpoglavij.

Četrto poglavje je posvečeno splošnonamenskemu procesiranju na grafičnih procesorjih. Izraz splošnonamensko procesiranje se v tem primeru ne sme ravno primerjati s tistim na osrednjem procesorju, saj je to mišljeno le v kontekstu, da so grafični procesorji dobili možnost računanja podatkov, ki niso strogo vezani na grafična izrisovanja. V enem izmed podpoglavij pa so predstavljena najbolj uporabljana okolja za razvoj aplikacij s paralelnim procesiranjem na grafičnem procesorju.

Peto poglavje opisuje načine hranjenja podatkov in probleme povezane s tem. S paralelnim procesiranjem namreč ne pridobimo veliko, če procesor večino časa čaka na dostavo podatkov. Hranjenje podatkov je predstavljeno tako s strojnega stališča (pomnilniške hierarhije in trajno hranjenje podatkov v računalniških sistemih), kot tudi s stališča programske opreme (hranjenje v različnih tipih podatkovnih baz).

V šestem poglavju so predstavljeni geografski informacijski sistemi, saj je v poglavju s testiranjem energetske učinkovitosti in hitrosti procesiranja eden izmed testov narejen tudi v brezplačnem sistemu GRASS. Slednji ima nek modul napisan za splošnonamenski in grafični procesor.

Sedmo poglavje predstavlja jedro diplomskega dela, saj zajema testiranje hitrosti procesiranja na sodobnih grafičnih karticah in večjedrnem osrednjem procesorju. Vsaka strojna komponenta je na kratko predstavljena s pojasnilom, zakaj je primerna za testiranje. Z dobljenimi podatki lahko bralec dobi predstavo o razmerjih hitrosti procesiranja istega problema (v paralelni izvedbi) na osrednjem in grafičnem procesorju. Med samim merjenjem sem naletel na zanimive ugotovitve in težave, ki so prav tako opisane v tem poglavju.

Zadnje (osmo) poglavje je kratek splošni komentar k primerjavi paralel-

nega procesiranja na osrednjem in grafičnem procesorju ter kakšne so napovedi razvoja na področju strojne opreme.

Poglavje 2

Zgradba in delovanje procesorjev

Centralne procesne enote (CPE) so na trgu prisotne najdlje in predstavljajo nepogrešljivi del vsakega računalniškega sistema. Grafične procesne enote (GPE) so se pojavile v osemdesetih letih prejšnjega stoletja, hitreje pa so se začele razvijati v devetdesetih letih. Do približno leta 2007 so bile grafične kartice namenjene le izrisovanju grafičnih elementov na zaslon, od takrat dalje pa je njihova arhitektura postala bolj fleksibilna in se tem koristna za procesiranja, ki so po naravi in algoritmih podobna tistim za izris grafike. Poglavje je osredotočeno na opis zgradbe sodobnih večjedrnih splošnonamenskih ter grafičnih procesorjev.

2.1 Večjedrni CPE

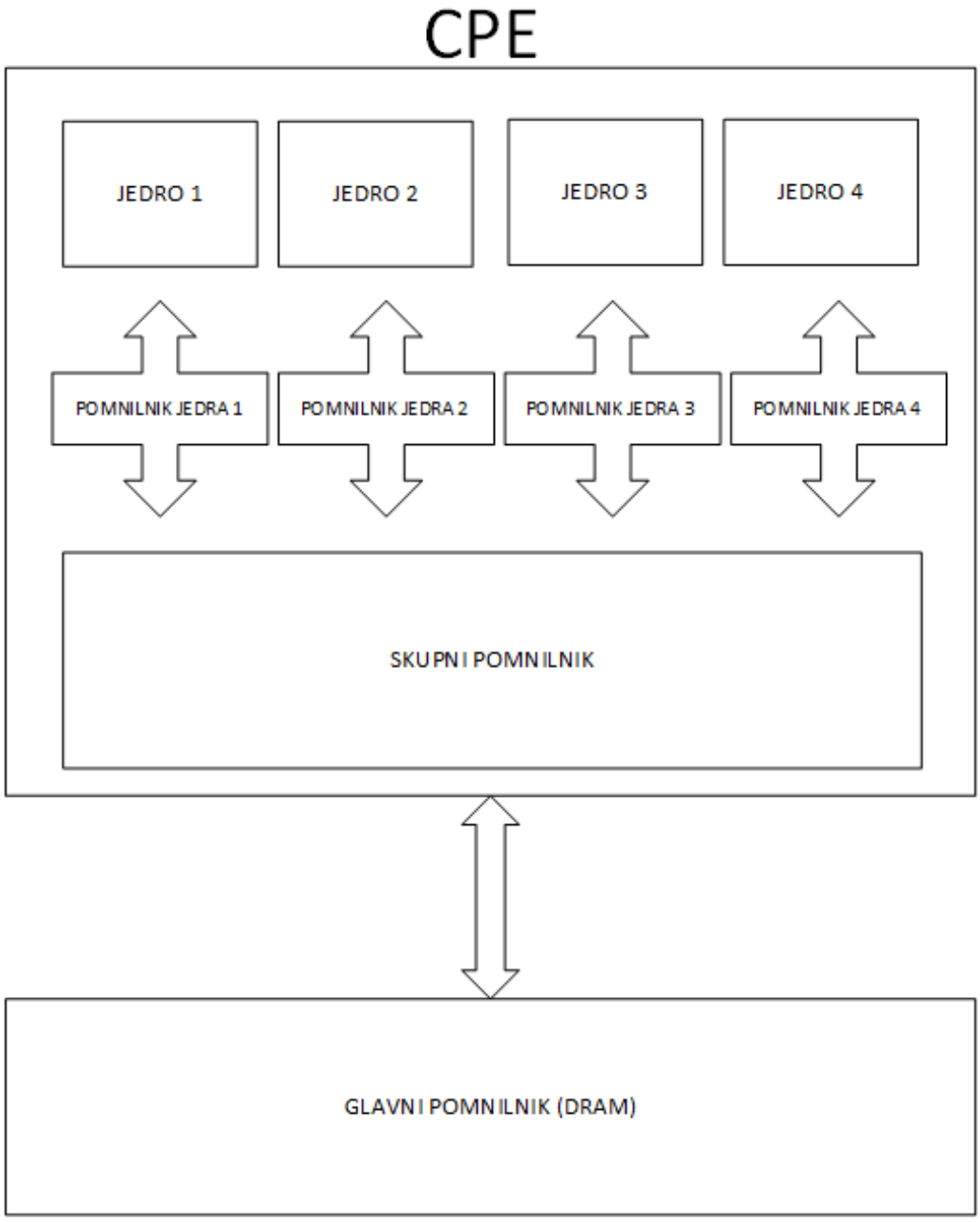
Po letu 2000 je na trgu frekvenca pojavljanja procesorjev (tudi najdražjih) začela močno upadati. Med procesorji arhitekture Intel x86 je kot enojedrni CPE imel največjo urno frekvenco Pentium 4 670 (kodno ime jedra Prescott), in sicer 3.8GHz. Intel je imel sicer v pripravi še različico s 4.0GHz, vendar ta ni nikoli ugledala luči sveta zaradi prevelikih težav z odvajanjem toplote. [3] Na tem mestu velja omeniti še dejstvo, da je podjetje Intel imelo smele načrte s serijo procesorjev Pentium 4 ob lansiranju na trg, saj so si obetali frekvence do 10GHz. [4] Tudi drugi proizvajalci hitrih CPE-jev so naleteli na

identične težave z višanjem frekvence ure, zato so po letu 2005 tudi v splošni rabi postali zelo popularni večjedrni procesorji.

Odločitev za večjedrne procesorje je izključno ekonomske narave, saj bi bilo povečevanje frekvence ure ekonomsko nesmiselno. Dobljena hitrost je ob porabi energije za nadzarovanje toplotnega segrevanja čipov precej zanemarljiva. Nasprotno pa je smiselno vgrajevanje večih identičnih sredic jeder znotraj enega fizičnega procesorja.Na tak način se je število procesnih enot sicer res hitro povečalo, vendar pa to ne vpliva nujno na hitrejše izvajanje neke aplikacije. V kolikor je znotraj enega procesorja več jeder, to navadno pomeni, da imajo le-ta tudi manjšo urno frekvenco. Aplikacije, ki so procesirale problem, ki je po naravi paralelen so tako hitro dobile posodobljeno verzijo, ki izkorišča več jeder, pri ostali aplikacijah pa pospešek ni bil opazen.

Ena od prednosti večjedrnih procesorjev je tudi, da se naenkrat izvaja več instanc programa, vsaka na svojem jedru. Na tak način se podatkov ne procesira hitreje, vendar samo več v enakem času, kar pa ni nujno vedno koristno. V naslednjih obdobjih se je število jeder, prav tako pa je hitro rasla tudi frekvenca ure pri Intel-ovih procesorjih. Intel je namreč saj je skupaj s prehodom na večjedrne procesorje prešel tudi na popolnoma prenovljeno arhitekturo čipov (imenovano Core). Arhitektura Netburst (Pentium 4) je imela predolge cevovode, da bi jo lahko še naprej učinkovito razvijali. Večjedrni procesorji pa žal s seboj prinesejo tudi kakšno težavo, ki je v dobi enojedrnih izvedenk ni bilo. Ena izmed takih je naprimer predpomnilnik (angl. cache) znotraj procesorja.

Običajno je hierarhija CPE realizirana tako, kot prikazuje slika 2.1. Vsako jedro ima svoj pomnilnik na najnižjem nivoju (imenovan L1), naslednji v organizaciji arhitekture pomnilnikov (L2 in ponekod še L3) pa so že skupni za celotni procesor. Ločeni predpomnilniki prinesejo težavo, saj se neko procesorsko jedro ne zaveda, kaj je shranjeno v L1 pomnilniku drugega jedra. Rešitev so usklajevalni mehanizmi, ki pa spet zahtevajo svojo realizacijsko logiko, ki pa je manj zapletena od rešitve, kjer bi L1 že od nivoja pomnilnika bila skupna vsem jedrom.



Slika 2.1: Tipična zgradba 4-jedrnega CPE procesorja

2.2 Zgradba GPE

Grafični procesor je v nekaterih pogledih grajen ravno nasprotno kot tradicionalni osrednji procesor v računalniškem sistemu.

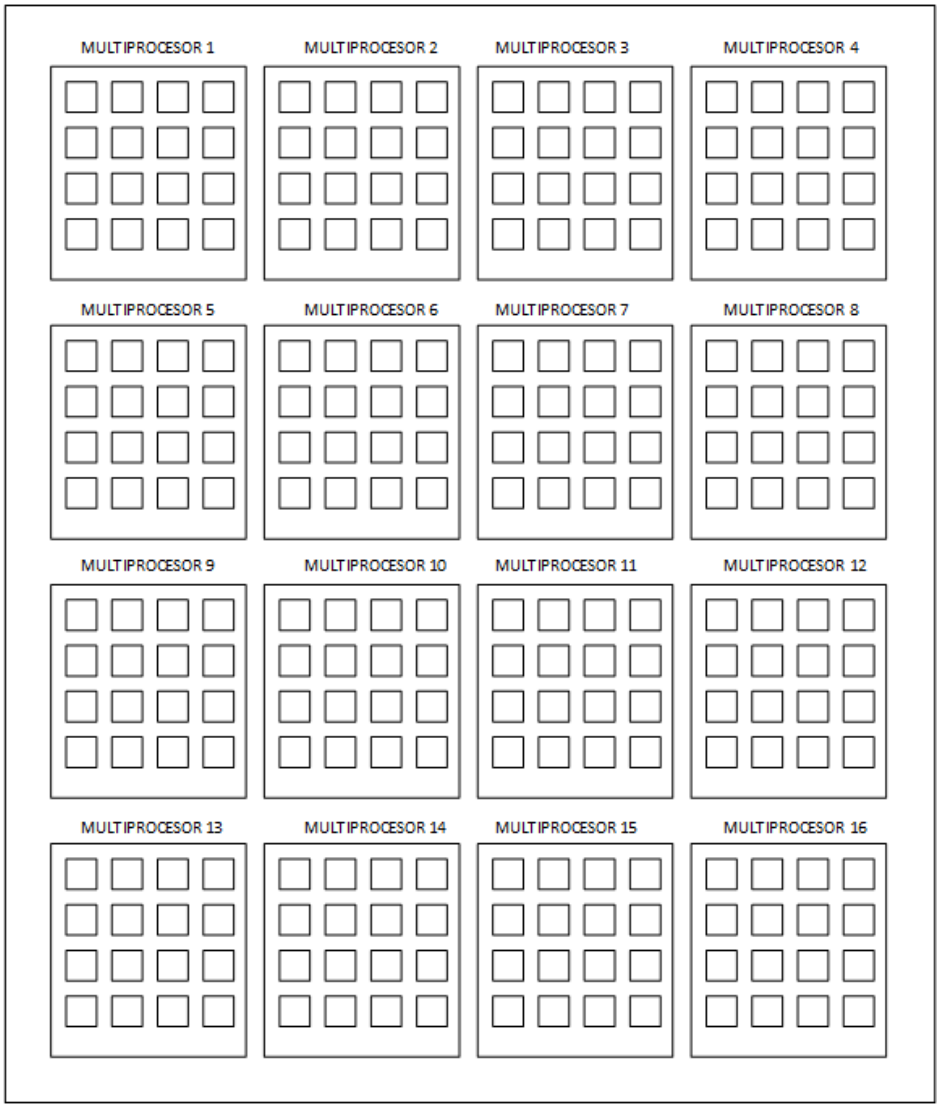
Osrednji procesor ima malo jeder (še ne tako dolgo nazaj samo enega), ki pa so hitra. CPE je z vidika programerja bolj primeren, saj tako odpadejo vsi problemi in (precej) težji algoritmi, ki so povezani z večimi procesorskimi jedri in dostopi do podatkov.

Grafični procesor je sestavljen iz več tisoč procesnih enot, ki so precej počasnejše od enega jedra v splošnonamenskem procesorju, vendar količina le-teh v določenih računalniških problemih kot odtehta hitrost enega samega splošnonamenskega procesorja (oz. več njih).

Več procesnih enot je združenih v en multiprocesor, ki se ga največkrat poimenuje s kratico SM, kar je okrajšava za Streaming Multiprocessor. Pri podjetju Nvidia tak multiprocesor znotraj grafičnega čipa označujejo kot SMX. Za poimenovanje termina SM še ni prevoda v slovenščino, vendar bi ga lahko prevajali kot *večprocesorska grafična enota*. Multiprocesor znotraj grafičnega čipa zmore naenkrat izvajati več blokov kode naenkrat, kar je pomembno, kadar se v programski kodi pripravlja klic za izvajanje procesiranja nad podatki preko grafičnega čipa. Zmoglivejše grafične kartice (čipi) imajo več multiprocesorjev.

Razvoj tehnologije v računalniški industriji je naletel na hudo težavo pregrevanja [5] in prekomerne porabe električne energije hitrih procesorjev. Zaradi vseh teh problemov se je razvoj računalniške strojne opreme (natančneje procesorjev) spremenil in se začel premikati v smer večjedrnega procesorja. S to spremembo je bilo mnogo dobrih algoritmov, ki so bili pisani za eno procesorsko jedro obsojenih na korenito spremembo ali pa celo načrtovanje popolnoma novega algoritma [6]. Za programerja je tak način manj sprejemljiv, saj je potrebno več časa posvetiti meritvam in testiranju izvajanja računalniškega programa na več procesorjih kot pa semantiki samega algoritma.

Na sliki 2.2 je prikazana zgradba sodobnega grafičnega procesorja, ki je v osnovi podobna pri vseh proizvajalcih teh procesorjev. Vsak grafični čip ima več enot SM, slednji pa več procesorkih jeder. Število SM-jev in količina jeder v njih se konstantno spreminja z novimi generacijami grafičnih procesorjev. Koliko procesorskih jeder bo vsebovala enota enota SM, koliko bo celotno število SM-jev v grafičnem procesorju in kakšna bi hitrost posameznih jeder znotraj SM-jev se odločajo pri dizajniranju nekega čipa. Proizvajalci pri razvoju tehnologije vedno iščejo najboljša razmerja, da dobijo v povprečju za vse vrste procesiranj čimboljše rezultate. V kolikor se procesorje izdeluje po naročilu, oz. se želi pospešiti določeno vrsto procesiranja, se število SM-jev, jeder in hitrosti le-teh prilagodi problemom, ki se jih bo reševalo.



Slika 2.2: Primer zgradbe grafičnega čipa

Poglavje 3

Paralelno procesiranje na CPE

Paralelno procesiranje v svetu računalništva pomeni hitrejše reševanje nekega problema, kjer pri enaki vhodni količini podatkov porabi neka funkcija manj časa da te podatke obdela in pretvori v izhodno informacijo. Poleg tega pa se za pojmom paralelnega procesiranja skriva tudi hitrost odziva računalnika uporabniku. Procesor namreč ni popolnoma zaposlen z izvajanjem operacij, zato del aplikacije, ki komunicira z uporabnikom, ni neodziven. Velja omeniti, da je zgoraj opisano funkcionalnost mogoče doseči že v računalniškem sistemu z enim procesorjem, ki ima eno fizično jedro. V tem primeru procesor preklaplja med različnimi opravili v zelo kratkih časovnih intervalih, kar pa je za uporabnika vseeno povsem transparentno. Dobimo vtis izvajanja večih opravil hkrati, kar v računalniškem žargonu poimenujemo *multiprogramiranje*.

Namen tega poglavje je predstaviti paralelno procesiranje na centralni procesorski enoti.

3.1 Paralelizem na nivoju bitov

Paralelizem na nivoju bitov spada na začetek razvoja paralelnega procesiranja. Pri tovrstnem procesiranju v nekih časovnih intervalih podvajamo dolžino računalniške besede, njegovi začetki pa segajo v leto 1970. Zatem je

sledilo obdobje, v katerem so pomen dobili daljši procesorski ukazi. Tako so 4-bitnim procesorjem sledili še 8, 16 in 32-bitni. Slednji so bili dolgo najpogosteje uporabljani, v zadnjem desetletju pa so se množično začeli pojavljati 64-bitni procesorji na arhitekturi x86, ki je nazaj kompatibilna. Pri tem velja omeniti, da gre za razširitev obstoječe arhitekture, poimenovane x86_64, ki jo je trgu najprej predstavilo podjetje AMD. Intel je imel pripravljeno svojo različico 64-bitne arhitekture, ki pa ni nikoli prišla na trg, saj je bila zaradi nekompatibilnosti s prvotno zelo razširjeno arhitekturo x86 nekonkurenčna AMD-jevi rešitvi. Omeniti velja, da se je Intelova različica 64-bitne arhitekture pojavila v procesorjih serije Itanium [7], ki pa nikoli ni zaživela v masovni proizvodnji. Omejena je bila namreč le na strežniški segment računalnikov, vendar se tudi tam ni uveljavila, saj za to procesorsko arhitekturo ni bilo pisanih programov.

Pomen paralelnega procesiranja na navoju bitov je, da z daljšanjem dolžine procesorske besede narašča število bitov, ki jih procesor lahko procesira v enem urinem ciklu. V praksi to pomeni, da 8-bitni procesor porabi več urnih ciklov za seštevanje 32-bitnih števil kot 32-bitni procesor, ki to isto nalogo opravi v eni periodi.

Zaenkrat se zdi, da je omenjeni način paralelnega izvajanje zaključil z razvojem, saj ne kaže, da bi se kdaj v srednji prihodnosti pojavili procesorji z daljšimi besedami. Prehod iz 32-bitnih na 64-bitne procesorje je bil bolj kot zaradi same hitrosti procesiranja opravljen zaradi omejitve naslavljanja količine pomnilnika (2^{32} = 4GB) in ker je taka količina v današnjem času cenovno zelo ugodna.

3.2 Paralelizem na nivoju procesorskih ukazov

Od sredine osemdesetih let prejšnjega stoletja, ko razvoj paralelizma na nivoju bitov ni več napredoval, so začeli raziskovanjem paralelnega izvajanja na nivoju procesorskih ukazov.

Takrat so bili že jasno prisotni procesorji s cevovodi [8]. Velik del raziskav

je bil usmerjen v to, kako s preureditvijo procesorskih ukazov doseči hitrejše izvajanje programa ob predpostavki, da se izvajanje programa navzven ne spremeni. (Ob istih vhodnih podatkih daje identičen izhod kot program, pri katerem ukazi niso urejeni v drugem (neoptimalnem) vrstnem redu). Vedeti je potrebno, da so bili takrat na trgu močno prisotni procesorji z dolgimi cevovodi (predvsem Intelova arhitektura) [9], pri katerih se je zaradi cevovodnih nevarnosti, in s tem čakanj ciklov urinih period, dalo s preureditvijo ukazov precej pospešiti izvajanje programa . Daljši kot je procesorski cevovod, več je lahko v nekem trenutku cevovodnih nevarnosti, kar pomeni da so s preureditvijo tudi pohitritve lahko večje. Na tak način se do neke mere lahko optimizira izvajanje programa, vendar kompleksnost algoritmov za preurejanje procesorskih ukazov pri (zelo) dolgih cevovodih raste zelo hitro. Procesor z enim od najdaljših cevovodov je bil Pentium 4 (31-stopenjski cevovod) [10], kar je v primerjavi z današnjimi procesorji serije Intel Core (12 do 14-stopenjski cevovod) [11] resnično precej daljše. Ob tem velja izpostaviti dejstvo da je možno z daljšimi cevovodi, in s tem preprostejšimi fazami znotraj ukaza, na precej lažji način dvigovati frekvenco procesorske ure.

Raziskave vzporednega izvajanja na nivoju procesorjev so se pojavile tudiz razlogom, da bi se vrstni red ukazov pri programih, ki so bili napisani za starejše procesorje, izvorne kode pa ni bilo na voljo, optimiziral za novejše arhitekture. Posebni primeri procesorjev, ki spadajo v to kategorijo vzporednega procesiranja so t.i. *superskalarni procesorji*, ki lahko izvajajo več ukazov hkrati, a samo, če so podatki med sabo neodvisni.

Današnji procesorji v večini primerov nimajo pretirano dolgega cevovoda, zato se so se pojavile nove študije kako izboljšati vzporedno izvajanje na procesorju.

Bistvo paralelizma na nivoju procesorskih ukazov je torej, da procesor v nekem trenutku izvaja več ukazov naenkrat in ne zaporedno enega za drugim. Omenjeni pristop pa je prinesel večje težave v procesorske arhitekture zaradi razveljavljanja že začetih ukazov [12].

3.3 Vzporedno procesiranje podatkov

Programske zanke predstavljajo velik del programov in zato izrazito vplivajo na čas izvajanja programa. Vzporedno procesiranje nad podatki velikokrat nastopa v primeru, ko se neka operacija zgodi nad več podatki hkrati. Taka vrsta procesiranja je v svetu računalništva znana pod kratico *SIMD (Single Instruction Multiple Data)*. Obstajajo še druge vrste vzporednega izvajanja operacij nad podatki, npr. hkratno izvajanje več ukazov na istimi podatki (*MISD – Multiple Instruction Single Data*) ter izvajanje več ukazov hkrati nad več podatki (*MIMD – Multiple Instruction Multiple Data*), ki pa niso tako razširjene kot SIMD.

Sodobni procesorji arhitekture x86 imajo SIMD zbirke ukazov kot so MMX, SSE itd., torej gre za več različic, ki so nastajale skozi čas, pri čemer vsaka različica vsebuje vse ukaze prejšnjih.

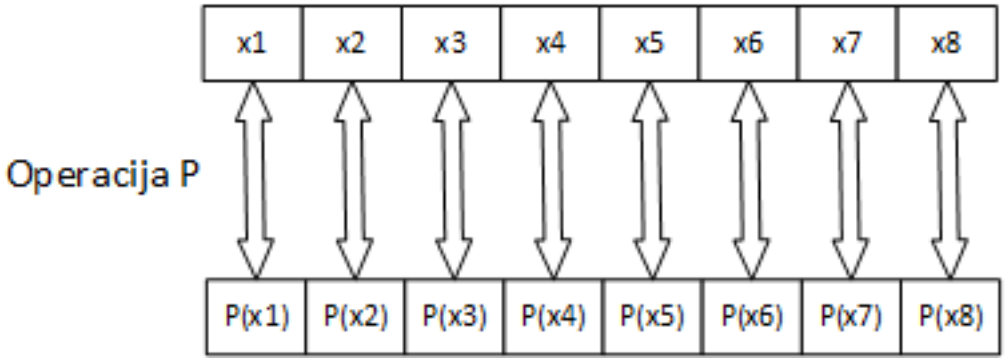
Primer programske zanke (v Microsoft C#), ki se jo da izvajati vzporedno:

```
int result[] = new int[100];
for(int i=0; i<100; i++)
{
    result[i] = i*2;
}
```

Očitno je, da med podatki ni nobenega prepletanja (kar je vizualno ponazorjeno tudi na sliki 3.1), zato se v takih primerih algoritem lahko izvaja vzporedno.

Spodnji primer prikazuje algoritem, v katerem nastopajo med seboj odvisne spremenljivke in ga zato ni mogoče izvajati vzporedno.

```
int number1 = 0;
```



Slika 3.1: Identična operacija, ki jo je kot tako možno izvajati vzporedno nad množico vhodnih podatkov.

```
int number2 = 3;
int sum = 0;
while(sum < 20)
{
    sum = number1 + number2;
    number1 = number1 + 2;
    number2 = number1;
}
```

Spremenljivka *number1* se mora najprej povečati (izračunati v programski zanki), preden spremenljivka *number2* dobi njeno vrednost. V vsakem obhodu **while** zanke pa morajo biti tudi vrednosti spremenljivk *number1* in *number2* enake pri vsaki ponovitvi programa. Ta zahteva pa ni avtomatično izpolnjena pri vzporednem izvajanju, saj operacijski sistem dodeljuje izvanjanje le-teh procesorju. Tako je od trenutnega stanja računalniškega sistema odvisno, katera programska nit se bo prej izvajala.

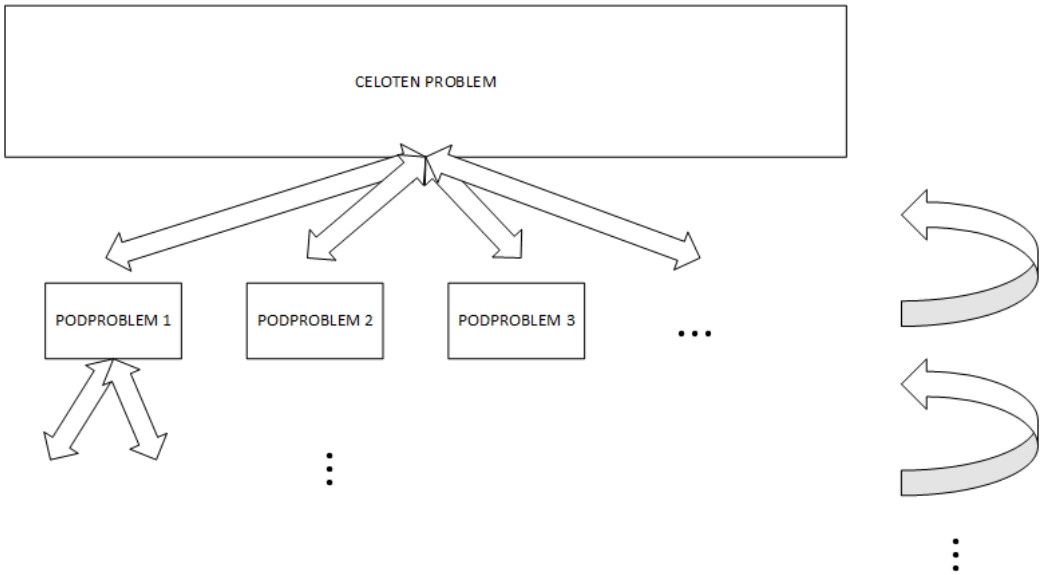
3.4 Vzporedno procesiranje več različnih opravil hkrati

Računalniški programski problem se lahko rešuje tudi na način, pri katerem se ga razbije na manjše, obvladljive dele (primer je prikazan na sliki 3.2), ki pa sami zase predstavljajo neko zaokroženo celoto. Tak pristop razbijanja na podprobleme je znan pod imenom *deli in vladaj (divide and conquer)*.

Tako razbita opravila (ang. task) lahko izvajamo vzporedno, če le niso pogojena na način, pri katerem se mora neko opravilo izvesti do konca preden se začne drugo. Podatki nad katerimi izvajamo tako vrsto vzporednega procesiranja, so lahko isti za vsa opravila ali različni.

Vzporedno procesiranje več različnih opravil hkrati (oz. angleško *task based parallelism*) je močno odvisen od problema, ki ga procesiramo. Algoritmi za reševanje programskih problemov, ki se jih bodisi ne da vzporedno procesirati (npr.: vhod nekega opravila je izhod prejšnjega in se naslednje opravilo ne more začeti preden se konča prejšnje) bodisi je težavnost takega algoritma precej prevelika in končna ocenjena pohitritev (največja možna, povprečna, najnižja) dopuščajo zelo malo možnosti za vzporedno izvajanje opravil.

Značilno za vzporedno procesiranje več (različnih) opravil hkrati je, da naraščanje količine vhodnih podatkov ne vpliva na hitrost programa na enak način kot se to dogaja pri nevzporednem procesiranju.



Slika 3.2: Slikovna ponazoritev razdelitve problema na podprobleme, ki so med seboj neodvisni.

3.5 Težave v povezavi s paralelnim izvajanjem

Prej omenjena ozka grla in velika razhajanja v hitrostih različnih računalniških komponent pridejo še močneje do izraza, če imamo v računalniškem sistemu več hitrih procesorjev, ki zmorejo vzporedno izvajati programsko kodo. Bralnih in pisalnih dostopov do komponent, ki hranijo podatke (diski, pomnilnik) je zato še mnogo več. Tudi to, da računalniške komponente, ki hranijo podatke niso bistveno hitrejše kot izvedenke za široko potrošnjo (npr. pomnilniki DRAM s samodejnim popravljanjem napak (ECC), ki so namenjeni strežnikom z visoko zanesljivostjo delovanja in navadnimi izvedenkami, ki omenjene lastnosti nimajo vgrajene), predstavlja težavo. Enako velja za magnetne trde diske. Trdi diski z vrtenjem 15.000 obratov na minuto sicer so hitrejši od trdih diskov za široko potrošnjo s 7.200 obrati, vendar pa se razlika dostopnih časov v primerjavi z DRAM-i še vedno meri v več stotisočkratni razliki.

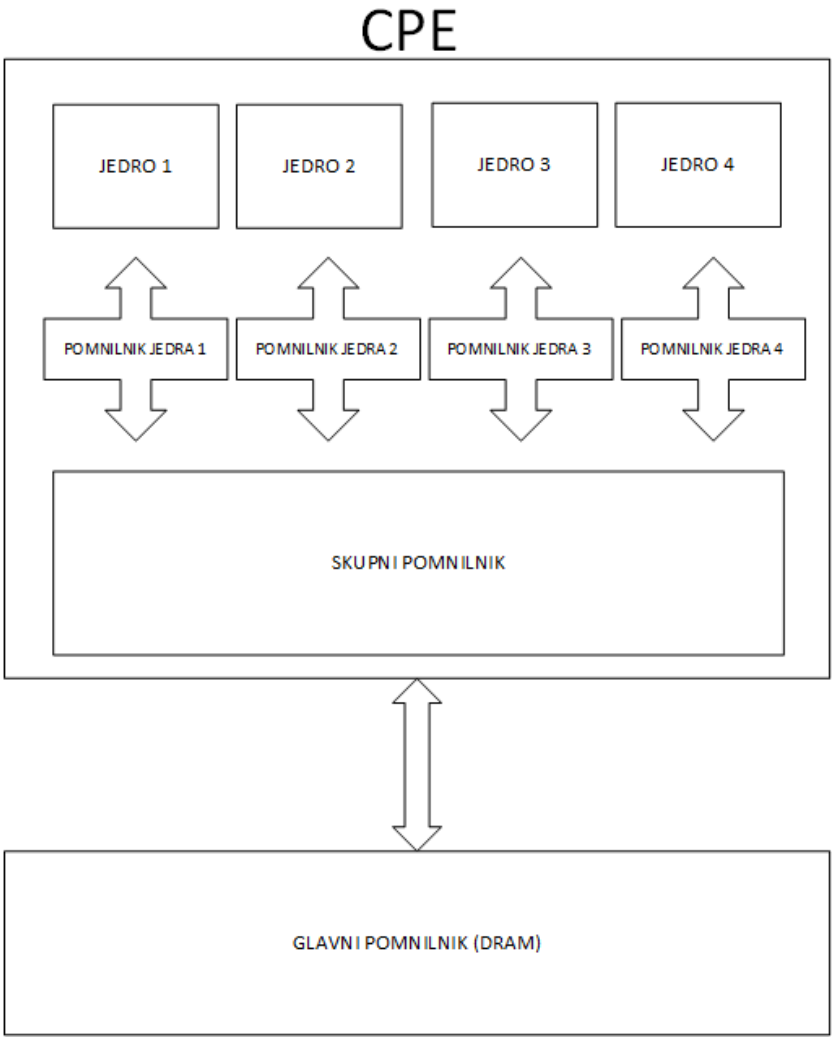
Tako velike hitrostne razlike je težko odpravljati ali jih z ustreznimi algo-

ritmi dostopanja izničiti. Eden on načinov reševanja omenjenega problema je pomnilniška hierarhija, ki je predstavljena na sliki 3.3.

Narava računalniške opreme za shranjevanje pa je taka, da je naprava zasedena, ko bere ali piše podatke. To povzroči še hujše ozko grlo v večprocesorskem (oz dovolj je že tudi en sam procesor z večimi fizičnimi jedri na eni silicijevi rezini) sistemu, saj se zahteve lahko generirajo z večih procesorjev istočasno. Slednji bodo tako še dlje čakali na zahtevan podatek oz. operacijo pisanja podatka.

Na tem mestu velja izpostaviti veliko prednost pomnilnikov DRAM in tudi diskov SSD z bliskovnimi celicami. Poleg več tisočkrat nižjega časa v primerjavi z magnetnimi trdimi diski, je še precej bolj enakomeren in predvidljiv (izračunljiv) čas dostopa. Čas branja in pisanja pri magnetnem trdem disku sta precej odvisna od trenutnega položaja glave in vrtenja plošče. Navaden (tovarniški) povprečen dostopni čas namreč predvideva, da se mora plošča v trdem disku zavrteti za pol obrata, kar se statistično sicer izkaže kot pravilno, vendar to nikakor ni nujno. Prav lahko se zgodi, da se mora plošča zavrteti za skoraj cel obrat. Nasproten primer je, kadar je (v sodobnih trdih diskih) informacija že v predpomnilniku in kontroler v trdem disku sploh ne izda ukaza za iskanje na magnetni plošči. Čas, ko dobi procesor podatek iz predpomnilnika diska, je znatno hitrejši, ni pa možno predvideti, kdaj se bo to zgodilo, saj je to popolnoma odvisno od prejšnjih in trenutnega stanja računalniškega sistema.

Upoštevanje vsega tega pri delovanju trdega diska vodi v zapletene algoritme optimizacije dostopa (npr. branja in pisanja). Za DRAM pomnilnike in flash celice je značilno, da je dostopni čas do vsake celice enako hiter, hkrati pa ima še prednost zaporednega dostopa do podatkov. Ti se namreč nahajajo na isti sledi na plošči diska in se berejo zaporedno, ko se disk vrti. Pri pomnilniških celicah sicer ni magnetne ploče, ki bi se vrtela, vendar pa se podatki zapisani v isti vrstici v čipu preberejo naenkrat. V računalniškem svetu je tak način znan pod imenom *burst mode (zaporedni način)*.



Slika 3.3: Slika prikazuje komunikacije med različnimi vrstami pomnilnikov v računalniški arhitekturi pri procesorjih z večimi procesorskimi jedri.

Poglavje 4

GPGPU in programska okolja

V sledečem poglavju je opisano eno izmed najhitreje razvijajočih se področij znotraj visoko zmogljivega računalništva (angleška kratica HPC), ki se je pojavilo okrog leta 2007. V računalniškem žargonu to področje poimenujemo *splošnonamensko procesiranje na grafičnih procesorjih*, najpogosteje pa se uporablja kar angleška kratica GPGPU. Predstavljena bodo tudi najbolj znana programska okolja za razvoj aplikacij, ki za procesiranje ne-grafičnih problemov izkoriščajo grafični procesor.

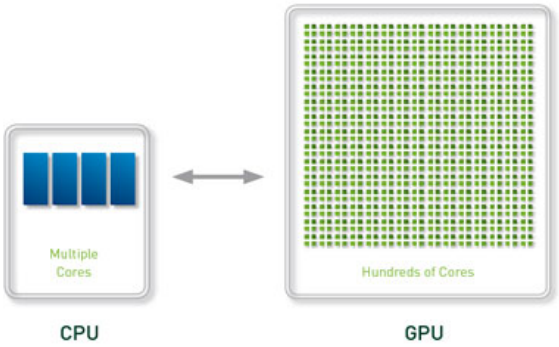
4.1 GPGPU

Grafične kartice oz. grafični čipi so na trgu prisotni že kar nekaj let - vse od osemdesetih let prejšnjega stoletja -, vendar so sposobnost izvajanja splošnega procesiranja dobili šele v zadnjem času. Primarno so te računalniške naprave še vedno namenjene delu z računalniško grafiko, npr. delu različnimi CAD programi za risanje načrtov zgradb, strojev in drugih podobnih objektov, delu z multimedijskimi vsebinami, obdelavi digitalnih video vsebin, upodabljanju (angl. rendering) grafičnih objektov, obdelavi digitalnih fotografij in vektorskih grafik, igranju računalniških iger, vizualizaciji podatkov ipd. Vse te tehnične zmožnosti pa so usmerjene v preozek krog uporabnikov. Največ uporabnikom, ki so v stiku z računalniki in niso strokovnjaki na enem

izmed teh področij, zadoščajo tudi najmanj zmogljivi grafični čipi. Ti so v času pisanja diplomske naloge največkrat integrirani kar v osrednjo centralno procesno enoto (CPE).

Grafični čip je sicer res procesor za posebne (specializirane) namene, vendar je sama zasnova takega procesorja, ki jo natančnejo opišemo v nadaljnih poglavjih, primerna tudi za ne-grafične računske probleme. To dokazujejo tudi programska orodja, pri katerih je del procesiranja prestavljen na grafične procesorje. Primer takih orodij so naprimer aplikacije podjetja Adobe [13]. S tehnološko zrelostjo proizvajalcev, pa tudi s trenutnim stanjem tehnologije v polprevodniški industriji, so se pojavili tudi ekonomski razlogi, da se razširi uporabo grafičnih procesorjev GPE (ang., Graphics Processing Unit). S tem se CPE razbremeni, poleg tega pa se izrazito pospeši izvajanje nekega programa oz. aplikacije. Kljub omenjenima prednostima procesiranje z GPE prihrani električno energijo, kar je pri stalno delujočih (velikih) računalniških sistemih zelo pomembno.s [14].

Grafični procesor je po zgradbi zelo podoben velikim superračunalnikom z ogromnim številom splošnonamenskih procesorjev (CPE), ki so skupaj povezani preko hitre računalniške mreže, le da je vse na eni sami kartici [15]. S tem do neke mere odpade skrb za mrežo, vsaj dokler ne začnemo s povezovanjem večih računalniških sistemov med seboj. Grafična kartica lahko nekatere programske probleme reši v enakem času kot računalniški sistem z večimi vozlišči. Vozlišče predstavlja en računalnik znotraj računalniške mreže. Komunikacijo med grafičnim čipom z velikim številom jeder in njegovim pomnilnikom (najbolj uporabljan je danes RAM tipa GDDR5 z efektivnimi hitrostmi do približno 6GHz [16]) lahko primerjamo s komunikacijo vozlišč (računalnikov) v velikem sistemu. Hitrosti prenosa podatkov znotraj take mreže se gibljejo od nekaj sto megabajtov do nekaj gigabajtov na sekundo pri najzmoglivejših mrežah [17]. Komunikacija med pomnilnikom na grafični kartici in njenim čipom pa danes znaša tudi do 300GB/s (tudi skoraj do 600GB/s, če ima grafična kartica fizično dva čipa na sebi) [18], torej govorimo od 2 do 3 velikostnih razredih razlike v hitrosti.



Slika 4.1: Primerjava zgradba splošnonamenskega procesorja (CPE) in grafičnega procesorja (GPE).

http://www.pny-europe.com/data/sitedynamic/Image/gpu-computing-feature.jpg

Nezanemarljiv podatek je tudi zakasnitev oz t.i. latenca pri komunikaciji. Navadno se zakasnitev znotraj lokalne mreže velikega računalniškega sistema meri v milisekundah [19], oz. pri mrežni opremi proizvajalca InfiniBand celo v mikrosekundah [20], dostopanje grafičnega čipa do njegovega RAM pomnilnika pa se meri v nanosekundah. Tudi tukaj je razlike za več velikostnih razredov, čeprav ne toliko, kot pri surovi hitrosti prenašanja podatkov v blokih. Na tem mestu velja omeniti primerjavo zakasnitve splošnonamenskega procesorja (CPE) do glavnega pomnilnika (ki je danes ponavadi tipa DDR). Zaradi nekoliko večje razdalje med CPE in pomnilnikom RAM ta zakasnitev danes tipično znaša med 40 do 50ns [22]. Sodoben CPE se odziva na prekinitve nekako med 10-15ns [21]).

Razlog, da so pri komunikaciji grafičnega čipa z njegovim pomnilnikom RAM zakasnitve večje, tiči v dejstvu, da je grafični čip namenjen procesiranju čimvečje količine podatkov v čimkrajšem času, četudi zaradi omenjene usmeritve čipa trpi odzivnost. CPE je s stališča odzivnosti narejen ravno obratno kot grafični čip. Hitrost odziva je velika, zakasnitve so torej nizke, vendar zaradi prekinitev ne premore velike prepustnosti podatkov.

Iz omenjene lastnosti grafične kartice sledi dejstvo, da mora programer pri razvoju in implementaciji algoritma skrbeti, da računanje nad podatki

poteka s čim manj skoki, saj to lahko precej zniža samo prepustnost oz. pretok podatkov.

Lastnost RAM-a\Tip RAM-a	**DDR3**	**GDDR5**
Podatkovni pretok	do cca 25GB/s	do cca 300GB/s
Cena	cca 4€/GB	višja
Podatkovna širina	Srednja (128 bit za dual channel)[12]	Visoka (256 bit ali več pri zmoglivejših karticah)[12]

Tabela 4.1: Primerjava nekaterih značilnosti osrednjega in grafičnega pomnilnika v sodobnih sistemih.

Največji problem nastane, kadar med izvajanjem aplikacije ugotovimo, da bi jo grafično procesiranje precej popešilo. Preproste in avtomatske pretvorbe, ki bi poskrbela, da bi že napisana aplikacija bila izvajana na grafičnem procesorju, ni mogoče narediti, saj se splošnonamenski procesorji precej razlikujejo od grafičnih, kar pomeni, da je tudi samo programiranje drugačno. Prav tako je potrebno za program, ki ima v ozadju nek algoritem, preveriti, če je le-ta še optimalen za izvajanje na grafičnem procesorju. Drugi problem, ki je prisoten že od samega začetka elektronskih računalnikov, je težava dovolj hitrega dostavljanja podatkov procesorjem, ki izvajajo operacije nad podatki. Razmerje med povečevanjem hitrostih procesorjev in napravami, ki hranijo podatke (tu so mišljene tako naprave, ki ob izgubi el. napajanja ohranijo podatek, npr. magnetni trdi diski, magnetni trakovi in v zadnjem času naprave z bliskovnim pomnilnikom), kot tudi take ki ob izgubi energije izgubijo vse shranjene podatke (npr. DRAM-i), se z leti poslabšuje. To pomeni, da se hitrosti procesorjem še vedno povečujejo hitreje kot napravam, ki hranijo podatke. To jasno nakazuje na težavo, da v računalniških sistemih, kjer ni posebno veliko težjih matematičnih izračunov[1] nad podatki, privede do čedalje slabše izkoriščenosti delovanja procesorjev.

[1]S pojmom težji matematični izračuni so mišljeni izračuni, kjer velik delež predstavljajo

4.2 Programska okolja za pisanje GPGPU aplikacij

4.2.1 CUDA (Nvidia)

Platforma imenovana CUDA (ang. **C**ompute **U**nified **D**evice **A**rchitecture) je programska platforma podjetja Nvidia za pisanje programske opreme, ki na enostavnejši način nudi dostop do grafičnih procesorjev in grafičnih kartic serije GeForce in Quadro.

Platforma se je v prvi končni različici pojavila že leta 2007, trenutno pa je na tržišču peta različica. Navadno z novo generacijo grafičnih čipov, ki so identični tako na karticah GeForce kot Quadro, pride tudi nova verzija CUDA platforme. Privzeti programski jeziki za pisanje v CUDA so C, C++ in Fortran, ki jih razširja in prireja podjetje Nvidia. Obstajajo seveda še programska okolja CUDA-o za druge programske jezike (npr. Java, Python, C#), ki pa niso razviti in podprti s strani Nvidie, vendar omogočajo uporabo grafičnih procesorjev v višjem programskem jeziku kot so C/C++ in Fortran.

CUDA je zaprta platforma, ki je omejena le na grafične procesorje podjetja Nvidia. Prav tako je trenutno dominantna platforma za razvoj programske opreme, ki izkorišča grafične procesorje za splošnonamensko računanje. Na tržišču se je pojavila prva in bila dobro marketinško promovirana.

4.2.2 OpenCL (Khronos group)

Programsko okolje OpenCL (**Open C**omputing **L**anguage) je v nasprotju s platformo CUDA popolnoma odprta in do tega trenutka že podprta od vseh večjih razvijalcev grafičnih čipov (AMD, Nvidia, Intel, ARM) tako v segmentu strežnikov in (grafičnih) delovnih postaj kot tudi na mobilnih platformah (tablični računalniki).

Za razvoj je zadolžena organizacija Khronos Group, ki je poznana predvsem po odprti knjižnjici za grafiko OpenGL. Organizacija Khronos Group

netrivialne računske operacije (npr. integrali, odvodi).

skrbi in razvija še druge odprte platforme (bolje rečeno, da skrbi za programske vmesnike) za multimedijske potrebe pri računalništvu.

Primarni programski jezik je tudi tu industrijsko uveljavljena in prirejena izpeljanka C-ja in C++-a. Uporaba OpenCL-a počasi a vztrajno narašča. Njegov razvoj zavira podjetje Nvidia, saj je OpenCL neposredna konkurenca rešitvi CUDA. Razvoj prevajalnikov za OpenCL pri Nvidii je precej neaktiven, kar je jasno razvidno iz dejstva, da zelo poredko posodobijo prevajalnike v svojih gonilnikih (imenovanih ForceWare).

Podjetje AMD je na omenjenem razvoju precej bolj dejavno, vendar v času pisanja veliko programerjev, ki programirajo v OpenCL platformi opaža nestabilno delovanje prevajalnikov [23], kar se jasno odraža v precej pogostih težavah pri prevajanju OpenCL kode. Prevajalnik se zaradi neznanih razlogov pri prevajanju pogosto sesuje, kot rečemo v računalniškem žargonu.

4.2.3 DirectCompute (Microsoft)

Microsoft-ov DirectCompute je del večje knjižnjice, namenjen delu z računalniško grafiko in je del DirectX-a od desete verzije dalje. Na voljo je le operacijskima sistemoma Windows in Windows Server od Viste (oz. Windows Server 2008) naprej.

DirectCompute podpirajo grafični čipi (kartice), ki podpirajo vsaj DirectX verzije 10, kar pomeni, da je zaenkrat pokrit le segment grafičnih delovnih postaj, namiznih računalnikov in prenosnih računalnikov.

4.2.4 Ostala programska okolja

C++ AMP (Microsoft)

Microsoft-ov C++ AMP (**A**ccelerated **M**assive **P**arallelism) je programska knjižnjica namenjena programerjem jezika C++. V njem razvijalci programske opreme na lažji način izkoriščajo paralelizem na nivoju podatkov. Sistem, na katerem se bo izvajala taka aplikacija, bo sam prevzel čimboljšega izvajnja paralelizma nad podatki. Knjižnjica je implementirana na osnovi DirectX 11

in omogoča, da osrednji procesor izvede programsko kodo, ki je grafični procesor ne more. Taka koda se optimizira tudi z ukazi tipa SSE, ki so značilni za osrednje procesorje Intel-ove arhitekture x86.

BrookGPU

Omenjeni prevajalnik je razvit na univerzi v Stanfordu in prav tako opravlja prevajanje in izvajanje programske kode na sodobnih grafičnih čipih podjetij Nvidia, AMD in Intel v operacijskih sistemih Microsoft Windows, Apple Mac OS in Linux. V praksi ni nikoli zaživel, saj je namenjen učenju programiranja na grafičnih procesorjih.

4.3 Izvajanje programske kode na GPE

Programska koda, ki je napisana za izvajanje na grafičnem procesorju, se ne začne izvajati samodejno, ampak mora osrednji procesor (CPU) najprej izvesti kodo, s katero naloži potrebne podatke na grafično kartico. Programska koda se tako deli na dva dela. Ta ločnica v kodi obenem predstavlja tudi ločnico med strojno opremo na kateri se del kode izvaja (oz. se bo izvedel).

Gostitelj

Gostitelj (angl. Host) je strojna oprema, ki izvede potrebne korake, da grafični čip dobi podatke nad katerimi potem izvaja računanje. Vlogo gostitelja prevzame osrednji procesor (CPE), ki izvaja t.i. serijsko kodo. To ne pomeni, da osrednji procesor ne sme imeti več jeder oz. da samo eno jedro znotraj procesorja izvaja serijsko kodo. Serijska koda je glavni del programa, ki vsebuje klice na grafični procesor.

Naprava (na kateri se procesira)

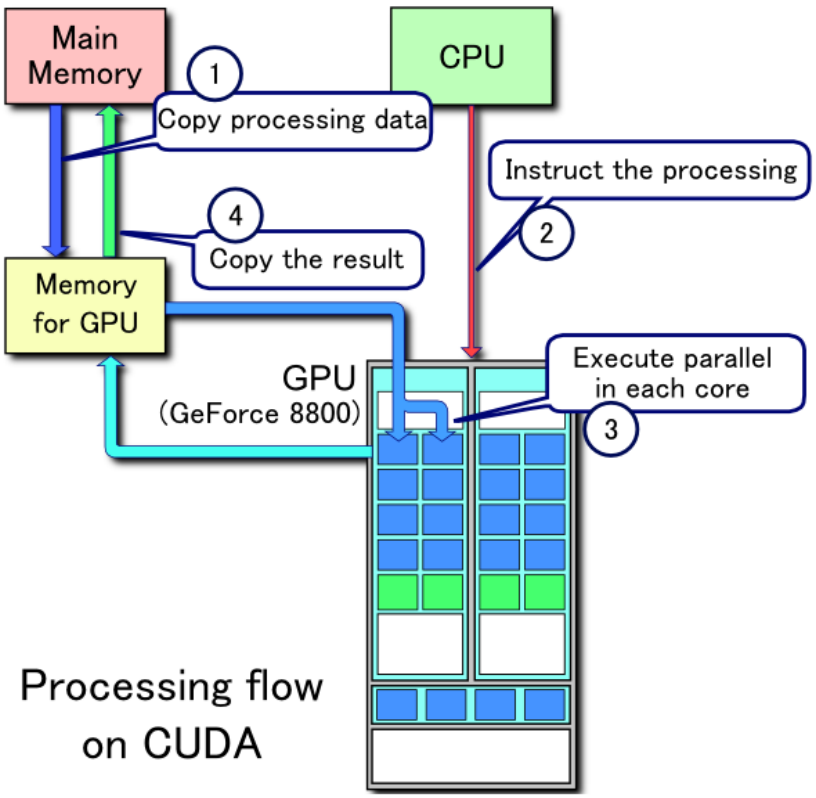
S pojmom naprava (angl. Device) je mišljena strojna oprema, kjer se bo izvajalo paralelno procesiranje. To je največkrat grafična kartica, ni pa nujno.

Zaradi intezivnega razvoja je vse pogostejša uporaba grafičnih čipov, ki se nahajajo kar v osrednjem procesorju. Ti so v času pisanja diplomske naloge že na stopnji surove računske moči, ki jo nudijo najcenejši modeli sodobnih grafičnih kartic (35-50€). V takem primeru sta host in device fizično v enem čipu.

Tako kot gostitelj izvaja serijsko kodo, naprava izvaja t.i. ščepec (angl. Parallel Kernel). Na gostitelju se lahko naenkrat izvaja natanko en ščepec. Če se v računalniškem sistemu nahaja več gostiteljev, torej več povezanih grafičnih kartic (npr. Nvidia SLI, AMD CrossFire), se lahko na vsaki grafični kartici istočasno izvaja ščepec. Vsak ščepec se potem znotraj grafičnega čipa izvaja na več nitih, kar je bistveno več, kot jih učinkovito zmore izvajati sodoben splošnonamenski večjedrni procesor.

Niti znotraj izvajanja v grafičnem procesorju so združene v skupine, mreža skupin (angl. groups) pa tvori t.i. paralelni ščepec (angl. parallel kernel).

Podobna terminologija je uporabljena tudi za OpenCL platformo.



Slika 4.2: Potek procesiranja s CUDA napravo

4.4 Efektivnost orodij za programiranje v GPGPU okoljih

Čeprav je nekatere serijske algoritme mogoče idejno zelo lahko prevesti v paralelen algoritem (npr. računanje oddaljenosti točk med seboj ali razdalja od neke referenčne točke), je veliko primerov tudi takih, kjer preprosta preslikava med zaporednim (serijskim) algoritmom in paralelno izvedenko ne obstaja. Včasih velika pohitritev sploh ni mogoča, saj je del algoritma nemogoče pretvoriti v paralelno obliko. To je t.i. Amdahl-ov zakon, ki ga v praksi ni mogoče preseči.

Problemi, ki jih zelo preprosto (celo trivialno) prevedemo v paralelno obliko, so v angleškem jeziku znani pod poimenovanjem *embarrassingly parallel*. V času pisanja diplomskega dela uradni prevod za omenjeno žargonsko

poimenovanje še ne obstaja, bi ga pa lahko opisali kot trivialno pretvarjanje algoritma v paralelno izvedenko. Ker so se GPGPU okolja za razvoj aplikacij v širši množični uporabi pojavila šelepar let nazaj, se še vedno močno razvijajo. Implementacija algoritmov, ki jih je težje prevesti v paralelne različice, še vedno zahteva nesorazmerno več časa in napora pri razvoju, kot pa je potem dejansko pridobljena hitrost končnega računanja oz. upravičenost iz kakšnega drugega ekonomskega stališča.

Tudi programerjev, ki so vešči pisanja in razvijanja aplikacij v GPGPU okolju, je v času pisanja diplomske naloge relativno malo, sploh v primerjavi s spletnimi programerji, in zato so temu primerno dragi. Na začetku razvoje vsake nove tehnologije so omenjene lastnosti razlog, da začne le-ta v splošno rabo pronicati počasi, saj stroškov dela ni mogoče v celoti naenkrat prenesti na produkt, preden bi ta začel povračati (gledano iz ekonomskega stališča).

Poglavje 5

Hranjenje podatkov

5.1 Lokacija hranjenja

Pri velikih računalniških sistemih, kjer se aplikacija izvaja na več računalnikih, se pojavijo enaki problemi, kot pri pomnilniški hierarhiji znotraj enega računalnika. Razlika je le v tem, da celoto kamor se podatki shranjujejo tvorijo skupaj. Tak prostor se največkrat poimenuje *pool* (ang.) ali deljeni podatkovni prostor (angl. distributed shared storage).

Omeniti velja, da deljeni prostor, ki ga uporabljajo vsi računalniki znotraj sistema, niiso nujno magnetni trdi diski. Zaradi vse večje uporabe podatkovnih baz, ki hranijo podatke v pomnilniku (angl. In-Memory Database), je deljeni prostor lahko tudi pomnilnik ali kakšna druga računalniška komponenta za shranjevanje podatkov. V primeru pomnilnika oz. komponent, ki ob izgubi napajanja izgubijo podatke, je nujno zagotoviti sistem neprekinjenega napajanja (angl. UPS). Druga možnost je da programska oprema sama skrbi, da se spremembe v deljenem pomnilniku odražajo tudi na napravah za shranjevanje, torej tudi ob izgubi napajanja obdržijo informacijo. Take naprave so navadno magnetni trdi diski in SSD-ji. V kolikor je uporabljen drugi princip shranjevanja, mora programska oprema za programerje in razvijalce na takem sistemu delovati transparentno. Dostop je možen samo do deljenega pomnilnika, medtem ko se spremembe samodejno shranjujejo npr.

še na magnetne trde diske.

5.2 Ozka grla

Z oddaljenostjo od procesorja postaja, podobno kot pri pomnilniški hierarhiji, počasnejša tudi komunikacija. Povezave med računalniki se ne merijo več v centimetrih, ampak v metrih ali celo večjih enotah. Ob vsem tem seveda naraščajo tudi dostopni časi do skupnih virov podatkov, zato se tudi tukaj izkorišča načelo časovne in prostorske lokalnosti podatkov. Podatki na nižjem nivoju v hierarhiji se sinhronizirajo z zamikom in ne takoj, ko se na katerem izmed nižjih nivojev zgodi sprememba stanja nekega podatka ali množica podatkov. Tako se prepreči prekomerna raba vodila in s tem povezane težave, predvsem enormno višanje odzivnega časa nad določenim odstotkom zasedenosti vodila. S pošiljanjem večih podatkov hkrati se namreč lahko izkorišča še ena lastnost, poznana s prenosov podatkov znotraj računalnika, in sicer t.i. neprekinjeni prenos (angl. burst mode), ki omogoča prenos velikega bloka podatkov z enim samim potrjevanjem. Tako se poveča izkoristek komunikacijskega kanala, saj večji delež komunikacije predstavljajo podatki in precej manjši delež potrjevanje pravilnosti prenosa.

5.3 Vrste podatkovnih baz

Najbolj razširjen tip podatkovnih baz so relacijske podatkovne baze, vendar še zdaleč niso edine. V zadnjih letih se predvsem zaradih družabnih omrežjih (npr. Facebook, Twitter...) pojavlja potreba po drugačnem tipu podatkovnih baz. Spremembe, ki jih uporabnik naredi morajo biti v istem trenutku vidne uporabnikom na različnih koncih sveta. Podatkovne baze, ki jih uporabljajo socialna omrežja in pripadajo skupini NoSQL nerelacijskih podatkovnih, sicer niso ves čas popolnoma sinhronizirane med seboj. To se pri objavi nekega uporabnika na socialnem omrežju vidi, ko del preostalih uporabnikov že vidi objavo, drugi pa jo bodo šele čez nekaj časa, navadno ekaj sekund ali nekaj

minut kasneje.

Omenjena funkcionalnost je za uporabnike sicer sprejemljiva, s tehnološkega stališča pa to pomeni precejšnje prihranke računske moči pri sinhronizaciji, saj slednje pri več TB velikih bazah lahko pomeni velik padec hitrosti aplikacije.

Podatkovne baze NoSQL so največkrat tipa ključ-vrednost (angl. key-value), kar pomeni da se pri poizvedovanju sklicijemo na ključ, da dobimo njegovo podatkovno vrednost. Taka zasnova baze se precej dobro, oz. bolje pri enaki strojni opremi glede na relacijski tip baze, obnese pri realnočasnem analiziranju ogromnih količin podatkov, kjer struktura samih podatkov ni tako pomembna. Zavedati se je potrebno, da stik tabel v relacijski podatkovni bazi predstavlja časovno potratno operacijo, ki skupaj s količino podatkov hitro narašča. V kolikor taka funkcionalnost pri neki aplikaciji ni potrebna, se s prehodom na kakšno od NoSQL baz ogromno pridobi na hitrosti poizvedb. [24]

Drug tip NoSQL baz, ki pridejo do izraza v GIS sistemih, so graf (angl. graph) baze. Kot je že iz imena razvidno, v takih bazah nastopajo vozlišča in povezave med njimi. Vsako vozlišče si lahko predstavljamo kot objekt z neko vrednostjo in z lastnostmi, ki so značilna za vozlišča grafa. Tudi povezave imajo definirane lastnosti, ki npr. povedo kolikšna je razdalja med vozliščema, ali pa npr. če imamo opravka s cestnim omrežjem, kakšen je naklon med vozliščema in kolikšna je dovoljena hitrost vožnje. Z graf tipom podatkovne baze NoSQL pa jasno nismo omejeni na strogo geografske relacije. Z grafom namreč lahko nazoorno predstavimo model družabnih omrežij, in ga tako uporabljajo npr. za prilagoditev marketinga in oglaševanja točno določenemu krogu ljudi.

Poleg same hitrosti med povezavami in objekti, je zapis v graf bazi tipa NoSQL za uporabo veliko bolj domač kot relacijske baze, predvsem za programiranje v objektno usmerjenih programskih jezikih.

NoSQL graf tip baze ima tudi Oracle-ov *Spatial and Graph*, ki je na voljo kot dodatek k njihovi relacijski podatkovni bazi.

5.4 Podatkovne baze v pomnilniku (*angl. In-memory database*)

S trenutnim stanjem polprevodniške tehnologije in razmerji med dostopnimi časi za različne komponente v računalniškem sistemu, so se pojavilev tudi podatkovne baze, ki se primarno nahajajo v pomnilniku. V primerjavi s trdimi diski imajo samo dve pomanjlkjivosti. Ena je ta, da ob izgubi napajanja pomnilnik izgubi podatek, česar si v računalniškem sistemu ne smemo privoščiti, zato so bodisi taki računalniki priklopljeni na stalno napajanje (angl. UPS) oz. se pomnilniku kako drugače zagotovi stalno napetost bodisi se v ozadju spremembe baze ves čas sinhronizirajo na napravo, ki ohrani podatke tudi ob izpadu električne napetosti - npr. trdi disk, čeprav ga vse pogosteje zamenjujejo naprave z bliskovnimi celicami.

Poglavje 6

GIS sistemi

6.1 O GIS sistemih

GIS je kratica za **G**eographic **I**nformation **S**ystem. Začetki GIS sistemov segajo v leto 1968, ko je angleški geograf dr. Roger F. Tomlinson objavil članek o geogfrafskem informacijskem sistemu za regionalno planiranje. Računalniška kartografija je sicer obstajala, vendar v drugačni obliki. Tomlinson je postavil temelje večplastne predstavitve geografskega področja (npr.: plast vegetacije, plast cestnega omrežj.), na osnovi njegove teorije pa je nastal prvi geografski informacijski sistem v Kanadi, ki se je imenoval CGIS. Sam je bil tudi vodja projekta, sistem pa so poganjali veliki centralni računalniki (angl. mainframe), saj je bila količina podatkov zaradi njihove večplastnosti zelo velika.

Sistem CGIS ni nikoli dosegel komercialne rabe, ampak je služil kot odličen model nadaljnemu razvoju tovrstnih sistemov. Plasti so postavljene nad koordinatni sistem. Takrat se še ni govorilo GPS koordinatah, te so namreč uvedli šele v sedemdesetih letih prejšnjega stoletja.

Po pojavu GPS-a, s katerim smo pridobili standarden koordinatni sistem za ves planet, se je v osemdesetih začelo pojavljati veliko GIS sistemov. Med njimi velja izpostaviti MIDAS, ki je bil prvi GIS sistem za takratni Microsoft-ov DOS. Kasneje je bil preimenovan v MapInfo in prirejen za še

bolj uporabljan operacijski sistem Microsoft Windows. Geografski informacijski sistemi so v večini primerov prirejeni potrebam stranke (podjetja), ki ga je kupilo oz. plačuje najemnino. Omenjeno dejstvo pravzaprav ni nobena posebnost, saj so večinoma vsi informacijski sistemi (npr. proizvodni IS, BI sistemi) prirejeni individualnim potrbam in posebnostim stranke.

V današnji dobi hitrih internetnih povezav, brezžičnega prenosa podatkov in majhnih računalnikov (mobilni telefoni, tablični računalniki), so se GIS sistemi še bolj uveljavili zaradi aplikacij, ki beležijo našo trenutno lokacijo in npr. izrisujejo pot, ki smo jo naredili v določenem času.

Ker je mobilnih naprav veliko in vsaka izmed njih ponavadi kominucira s strežnikom podjetja, ki je aplikacijo razvilo, mora biti računalniški sistem zares strojno zelo zmogljiv in imeti široko podatkovno povezavo s spletom, da lahko zadosti vsem potrebam sprejemenja, oddajanja in analiziranja podatkov.

6.2 Zahteve po strojnih računalniških komponentah pri sistemih GIS

Po pregledu strojnih zahtev za namestitev GIS sistema (oz. kot dodatka k obstoječi bazi) sem samo pri aplikaciji Manifold zasledil, da zna izkoriščati sodobne grafične kartice preko ogrodij za splošnonamensko računanje na grafičnih karticah (GPGPU). Omenjena aplikacija zna izkoriščati CUDA jedra na grafičnih procesorjih podjetja Nvidia.

Pri ostalih produktih (med drugim tudi ArcGIS, ki trenutno velja za enega največjih GIS sistemov) pa ni nikjer omenjeno, da s primerno grafično kartico oz. čipom pri nekaterih računskih operacijah dosegajo bistveno hitrejše procesiranje podatkov. Razlog je verjetno, da se tako veliki in obsežni sistemi, kot si GIS težje prilagajajo novih tehnologijam v strojni in programski opremi. Podjetja, ki razvijajo takšne sisteme, so namreč bolj konzervativna in se raje poslužujejo že dobro uveljavljenih in zanesljivih tehnologij, ki bodo zagotovo še nekaj časa prisotne na tržišču.

Obdobje splošnonamenskega procesiranja na grafičnem čipu se je začelo okrog leta 2006, ko je Nvidia izdala prvo končno verzijo razvojnega okolja CUDA. Od takrat je sicer minilo že nekaj let, vendar se v komercialnih produktih programske opreme uveljavlja šele v zadnjem letu ali dveh. OpenCL je bil v prvi končni različici lansiran na trg šele leta 2009, zato je šele v zadnjem letu začelo malo hitreje naraščati število aplikacij, ki izkoriščajo zmogljivost grafičnega čipa preko tega odprtokodnega ogrodja.

6.3 Projekt CudaGIS

CudaGIS je projekt, ki se v času pisanja diplomske naloge intenzivno razvija. Javno dostopna različica aplikacije sicer še ni na voljo, vendar so razvijalci že demonstrirali posamezne dele funkcionalnosti. Namen projekta je razdeliti GIS sistem na vsebinsko zaokrožene celote (module) in potem te module postopno prevajati iz zaporednih (serijskih) algoritmov v vzporedne. Sodelujoči v projektu so že demonstrirali obetavne pospešitve izvajanja nekaterih operacij na GIS sistemih. Implementacije algoritmov v CUDA (Nvidia) okolju so v primerjavi z implementacijo na navadnih splošnonamenskih procesorjih (CPU) hitrejše v povprečju od 10 do 40-krat, podatki pa so hranjeni na navadnih magnetnih trdih diskih.

Če so podatki in podatkovne strukture hranjene v pomnilniku RAM, so hitrostne razlike izvajanja med grafičnimi čipi (GPU) in splošnonamenskimi čipi (CPU) celo od 1000 do 10.000-krat hitrejše in tako dobre rezultate razvoj tehnologije le redko doseže.

Nikjer sicer ni zapisano koliko in katere splošnonamenske procesorje ter grafične kartice so uporabili, da bi se izračunalo največjo možno porabo električne energije in s tem izmerilo energetsko učinkovitost računanja. Smisleno je predpostaviti, da je konfiguracija grafičnih kartic med delom porabljala več energije kot konfiguracija s splošnonamenskimi procesorji. Vendar kljub nekaterim t.i. *embarrassingly parallel* pretvorbam algoritmov je mogoče verjeti, da so za isto opravljeno delo grafične kartice porabile bistveno manj energije

za delovanje.

6.4 Projekt GRASS GIS

Projekt GRASS GIS je eden najbolj popularnih GIS sistemov, ki pa so še zaščiteni z GNU licenco, torej je programsko opremo mogoče do določene mere opredeliti kot odprto. Začetki segajo že v leto 1984, kar še dodatno potrjuje kvaliteto, ki jo ima sistem še danes.

Ena močnejših strani GRASS sistema so nedvomno dodatki (ang. Addons), ki jih lahko razvija vsak, poleg tega pa so voljo širši javnosti. Sistem modulov je pregleden, saj je razdeljen na rasterski in vektorski del, ki sta lahko predstavljena v dveh ali treh dimenzijah. V podmapi raster se tako nahajajo programski moduli za delo z 2D rasterskimi mapami in slikami, v mapi raster3d programski dodatki za delo s 3D grafiko, v mapi vector pa moduli za obdelavo informacije v vektorski obliki. Programski moduli so večinoma spisani v programskih jezikih C, C++ in Python. Slednji je popularen skriptni jezik, ki se ga v kakšnem izmed programskih ogrodij (angl. framework) uporablja tudi pri sodobnih spletnih aplikacijah. Tudi poimenovanje modulov je precej dobro, saj se mape za posamezen modul začnejo s primernim začetkom okrajšave (za raster r, za 3D raster r3 in za vektor v), ki mu sledita pika in ime funkcije. Na tak način se lahko kasneje možno razviti paralelno sprogramirane programske izvedenke modula. V mapi raster tako obstaja modul compress (r.compress), ki kompresira in dekompresira rasterske zemljevide. Če bi kdo želel implementirati to funkcionalnost, da bi izkoriščal CUDA tehnologijo grafičnih čipov podjetja Nvidia, bi lahko naprimer naredil programski modul r.compress.cuda. Ostali uporabniki bi tako lahko uporabili knjižnico, če bi le njihov računalniški sistem razpolagal s primerno strojno opremo. V tem primeru bi to seveda bila dovolj sodobna grafična kartica podjetja Nvidia).

Omenjeni pristop k prepisovanju strojne opreme ni primeren le za sisteme tipa GIS, ampak tudi za ostale, saj je potrebno sprogramirati oz. doprogra-

mirati le tiste module, ki izrazito doprinesejo k hitrosti samega izvajanja. Šele po realizaciji le-teh se doprogramira module, ki manj doprinesejo hitrosti. Naslednja prednost takega pristopa je tudi možnost izbire primerne izvedenke za uporabnikov računalniški sistemi, t.j. izbira lahko kodo, ki se izvaja serijsko ali paralelno paralelno). Seveda se da primernost strojne opreme, v našem primeru je to zmožnost izvrševanja CUDA in/ali OpenCL kode, ugotavljati tudi na programski način in uporabniku predlagati ustrezne nastavitve.

Tretja dobra lastnost pristopa pa je, da razvoj modulov s serijsko kodo ne vpliva na razvoj modula s paralelno kodo. Projektu tako ni potrebno stati, da se velik del aplikacije prepiše v paralelne module, če je to slučajno potrebno.

6.5 Splet in paralelno procesiranje na klientu

Spletni brskalniki so v zadnjih letih dobili javascript kode, ki so za red velikosti hitrejše od prejšnjih. Prav tako se je močno pohitril in razvil CSS (ang. Cascading Style Sheets), ki ga uporabljamo za oblikovanje. Aplikacije, ki niso preveč zahtevne za klienta oz. intenzivne za izvajanje, hitro preselile v spletne brskalnike, kjer lahko potekajo tudi samo lokalno znotraj mreže. S tem operacijski sistem nameščen na računalniku ni več predstavljal skrbi.

Procesno intenzivne aplikacije, predvsem tiste za delo z grafiko (npr. GIS sistemi), še vedno ne morejo teči znotraj brskalnika, ki v takih primerih deluje kot programska oprema in s tem posnema računalniško neodvisno platformo. Programiranje za splet namreč ne nudi dovolj nizkega dostopa do strojne opreme, ki je nujen, da se algoritni ustrezno optimizirajo in da med programsko opremo in strojno opremo ni preveč abstrakcijskih plasti. V bližnji prihodnosti se to lahko spremeni, saj nekatera velika podjetja (npr. Nokia, Mozilla, Samsung in Motorola) razvijajo odprt standard, ki je poznan pod imenom WebCL in pri katerem gre za povezovanje OpenCL platforme z javascript jezikom. Javascript koda se bo tako lahko preko OpenCL okolja

izvajala na grafični kartici. Povezava med grafično kartico in javascript-om pa bo grafični gonilnik, ki bo skrbel za pretvorbo javascript-a v OpenCL kodo.

Vsi sodobni in odprti spletni standardi, ki se pojavljajo z namenom, da spletni brskalnik za določeno funkcionalnost ne bi več potreboval namenskega vtičnika (ang. plug-in), ampak bi že sam spletni brskalnik vseboval kot del svojega jedra novo funkcionalnost, delajo samostojno. Takšna je tudi filozofija WebCL-a. V času pisanja diplome tudi najzgodnejše preizkusne in predogledne različice najbolj priljubljenih spletnih brskalnikov (Google Chrome, Mozilla Firefox, Microsoft Internet Explorer, Apple Safari in Opera) omenjenega standarda še ne podpirajo. Na voljo so le preizkusni vtičniki, ki se jih težko dobi, in demonstracijski videi, ki prikazujejo različne simulacije npr. mešanja tekočin z navadnim javascript-om in javascript-om, ki uporablja OpenCL. Rezultati so presenetljivo dobri, vendar po pregledu omenjenega področja sklepam, da bosta pretekli še najmanj dve leti, preden se bodo pojavili brskalniki, ki bodo imeli v sebi integrirano zmožnost WebCL platforme.

WebCL bi lahko pomenil edega od večjih prelomov (ne pa tudi edinega možnega), da se GIS sistemi iz programske opreme, ki je vezana na operacijski sistem, preseli v spletni brskalnik. Problematična bi lahko bila količina podatkov, ki se mora prenašati po mreži. Serverji bi sicer lahko bili bolj razbremenjeni, saj bi podatke brez obdelave le poslali klientu, vendar bi količina le teh ob počasni povezavi povzročila dolgotrajno čakanje.

Čez nekaj let bodo tudi mobilne naprave dovolj zmogljive, da bodo zmogle preračunati ogromno količino podatkov (recimo nad velikim rasterskim zemljevidom). Kompresija bo v takih primerih ključna in se bo morala izvajati tako na strani strežnika kot na strani klienta, če bo šlo za dvostransko izmenjavo podatkov. Algoritmi za kompresijo so povečini serijske narave in jih ni možno hitro in učinkovito izvajati na grafičnem čipu. Kljub temu zna ovira postati pomnilnik, saj se ga pri kompresiji intenzivno naslavlja in je pri mobilnih napravah (mobilnih telefonih, tabličnih računalnikih) zaradi varče-

vanja z energijo precej počasen glede na zmogljivost in razvitost grafičnih čipov.

Poglavje 7

Testiranje energetske učinkovitosti procesiranja

7.1 Testni računalnik

Testni računalnik je bilo potrebno sestaviti premišljeno, saj se poraba energije med splošnonamenskimi procesorji (CPE) in grafičnimi karticami (GPE) precej razlikuje. Najzmoglivejše med grafičnimi karticami imajo navadno maksimalno energetsko porabo med 250W in 300W (npr.: Nvidia GeForce GTX 690 ima 300W.), najzmoglivejši procesorji za namizne računalnike pa le okrog 95W (npr. Intel Core i7 3770K). Serverske izvedenke procesorjev so lahko tudi močnejše, nekje do 150W, zelo redko več.

Primerjava energetske varčnosti ne bi bila smislena, če bi grafična kartica lahko porabljala bistveno več energije kot osrednji procesor. To je prva od dveh ključnih stvari, ki sem ji namenil večjo pozornost. Največjo deklarirano porabo čipa ali kartice se v svetu označuje s kratico TDP (ang. Thermal Design Power). Drugi pomemben dejavnik je starost samega čipa. Pri testni računalniški konfiguraciji sem bil pozoren, da sta grafična kartica in osrednji procesor na trg prišla približno istočasno.

Testni računalnik je imel sledeče strojne komponente:

Matična plošča: Gigabyte GA-H77M-D3H (vezni nabor Intel H77)
Osrednji procesor: Intel Core i5 3570K
RAM: Team Group DDR3 1600MHz, 11-11-11-28, 8GB (2x4GB), dual channel
SSD: Samsung 840 Basic 256GB, SATA 6Gbps
Grafične kartice:
- Nvidia GeForce 210 - Nvidia GeForce GTX650 - Nvidia GeForce GTX680

7.2 Razlaga izbire komponent

Komponente sem izbiral po kriterijih, ki so navedena v prejšnjih odstavkih ter bodo v naslednjih podpoglavjih obrazložena.

7.2.1 Matična plošča

Matične plošče se v računalniških sistemih ne razlikujejo po hitrosti, ampak predvsem po številu raznih priključkov, kot so npr. število priklopov SATA, število rež PCIe (PCI Express), število rež za pomnilnik,.... To velja za manjše serverske sisteme, omenjeno dejstvo pa potrjujejo tudi vezni nabori, katerih je vsaj na področju namiznih računalnikov precej manj kot pred desetletjem.

Vezni nabor Intel H77 na matični plošči Gigabyte GA-H77M-D3H spada poleg čipovja Intel Z77 v najvišji zmogljivostni razred za Intel-ove osrednje procesorje serije Intel Core i3/i5/i7 3xxx (kodno ime za to generacijo procesorjev je Ivy Bridge). Omenjena plošča je med najcenejšimi s tem veznim naborom, kar znese v času pisanja diplomske naloge približno 85€. Najdražje plošče z omenjenimi nabori lahko stanejo tudi 250€in več, kljub temu pa rezultati zaradi pasovnih širin vodil, ki jih čipovje podpira, na dražjih modelih ne bi bili nič boljši.

7.2.2 SSD

Namesto klasičnega magnetnega trdega diska sem se za shranjevanje podatkov odločil kar za disk s flash celicami, ki ima dostopne čase do podatkov veliko bolj konstantne, poleg tega pa so za več velikostnih razredov nižji. Tudi pretok podatkov pri sodobnih SSD pogonih je nekajkrat večji kot v klasičnih trdih diskih. SSD pogoni so cenovno že dostopni, zato jih imajo vgrajeni tudi računalniki srednjega cenovnega razreda. Mape (zemljevidi) v GIS sistemih so praviloma velike datoteke, zato sem pred testiranjem bil mnenja, da je predvsem zaradi cenovne dostopnosti smiselno uporabiti SSD.

7.2.3 Osrednji procesor (CPU)

Procesor Intel Core i5 3570K predstavlja najbolj zmogljiv procesor serije i5, ki ima tovarniško deklariran toplotni pečat (TDP) 77W. Tudi najmočnejši procesor s kodnim imenom Ivy Bridge (to je Core i7 3770K) ima prav tako deklarirano največjo porabo 77W, vendar je programski modul r.los sistema GRASS pisan samo za eno procesorsko jedro in tako večjih razlik med omenjenima procesorjema ne bi bilo. Prednost Core i7 3770K je predvsem v tem, da podpira HyperThreading (ima 4 fizična procesorska jedra in 8 navideznih), Core i5 3570K pa ima 4 fizična jedra, a ne podpira tehnologije HyperThreading in ima zato tudi samo 4 navidezna jedra. Tudi frekvenci procesorjev sta praktično identični, saj ima Core i5 frekvenco jeder 3.4GHz, Core i7 pa 3.5Ghz.

Kot sem že omenil, tehnologija HyperThreading pri serijski kodi r.los ne bi prišla do izraza, in zato hitrejše izvajanje pri isti deklarirani energetski porabi ne bo vplivalo na končne rezultate.

Sodobni procesorji imajo skoraj vsi že integriran grafični čip, ki se po hitrosti že lahko kosa s sodobnimi grafičnimi karticam, ki so v cenovnem razredu okoli 50€. Integriran grafični čip pri svojem delovanju izkorišča sistemski pomnilnik (npr. DDR3) in ne namenskega kot naprimer grafične kartice. S stališča pretoka podatkov je tako sicer omejen, vendar se ga pri

višjih prenosnih hitrostih navadno ne uporablja za grafično zahtevne naloge. Procesorji Intel s kodnim imenom Ivy Bridge so na trg začeli prihajati sredi leta 2012. Omeniti velja, da je bil pri tej platformi narejen prehod iz 32nm procesa izdelave na 22nm.

7.2.4 Grafične kartice (GPU)

Pri testiranju sem uporabil več grafičnih kartic, ki so se med seboj razlikovale po hitrosti pomnilnika (GDDR2, GDDR3 in GDDR5), največji energetski porabi in času prihoda na trg. Seveda so se razlikovale tudi v velikosti pomnilnika, ki pa pri majhni količini testnih podatkov nima vpliva.

Nvidia GeForce 210

Kartica GeForce 210 je že v času izida (konec leta 2009) namenjena cenovno najnižjemu razredu grafičnih kartic (cca 30€). Ime le 512MB počasnega GDDR2 RAM pomnilnika (1000MHz efektivno), 16 CUDA jeder s frekvenco 589MHz) in 64-bitno širino za komunikacijo s pomnilnikom.

Čip je narejen v 40nm tehnologiji in ima deklarirano največjo porabo 30.5W.

Nvidia GeForce GTX650

Omenjeno grafično kartico sem izbral, ker temelji na arhitekturi s kodnim imenom Kepler. Četudi je bila izdana jeseni leta 2012 je arhitektura čipa identična ostalim čipom iz serije Kepler, ki so izšli v prvi polovici omenjenega leta, kar dobro sovpada z izidom platforme Intel Ivy Bridge.

Prav tako kot pri Ivy Bridge, je tudi pri Kepler arhitekturi bil narejen prehod iz 40nm procesa izdelave na 28nm. Specifikacije modela GTX650 tehnično gledano niso nič posebnega, saj gre za grafične kartice, ki so ob izidu stale približno 110€.

Omenjeno kartico sem izbral, ker ima enake strojne karakteristike kot model GT 640 (enako število CUDA jeder, enako široko podatkovno vodilo

do RAM-a, enako število enot SMX), le da ima višje frekvence in ima RAM tipa GDDR5 in ne DDR3. Ustreznega modela kartice GeForce GT640 (na trgu je več revizij, ki pa se med seboj razlikujejo!) nisem uspel dobiti, zato sem se odločil, da uporabim kar model GTX650, ki ne porablja bistveno več električne energije kot model GT640. Omenjeni čip ima 384 jeder CUDA frekvence 1058MHz in GDDR5 pomnilnik frekvence 5000MHz.

Omeniti velja, da je bil RAM tipa GDDR5 (do efektivne frekvence 6008MHz) prisoten že pred letom 2012, zato ne poslabša ali izboljša situacije v prid boljšim rezultatom na grafičnih karticah, če delamo primerjavo na podlagi prihoda na trg.

Nvidia GeForce GTX680

Kartica je predstavljala najmočnejši čip omenjenega podjetja. Rezultati testov s to kartico služijo le kot podatek, kaj je bilo v letu 2012 najhitrejše za procesiranje aplikacij v okolju CUDA.

Čip ima 1536 jeder CUDA frekvence 1006MHz, pomnilnik tipa GDDR5 efektivne frekvence 6008MHz in 256-bitno povezavo s pomnilnikom. Najvišja deklarirana poraba je 195W.

Cena ob izidu se je gibala okoli 500€.

7.2.5 Poraba integriranega grafičnega čipa v Core i5 3570K

Proizvajalec (Intel) ne navaja, koliko energije lahko največ porablja v procesor integriran grafični čip. Naveden je le podatek o največji skupni porabi grafičnega čipa in osrednjega procesorja.

Z aplikacijo HWiNFO se da na empiričen način dobiti približno informacijo o omenjenih podsklopih celotnega procesorja. Celoten procesor se obremeni z namensko aplikacijo, ki popolnoma obremeni tako osrednji procesor kot integrirani grafični čip.

Vsota podatkov je zato blizu navedene največje porabe celotnega pro-

cesorja, ki je za ta procesor 77W. Integrirani grafični čip se da izklopiti v BIOS-u matične plošče, kar sem tudi storil, saj sem v računalniku vedno imel eno ločeno grafično kartico.

Smiselno se mi zdi, da se pri energetski primerjavi maksimalno porabo integriranega grafičnega čipa odšteje, saj predstavlja kar velik delež celotne porabe. Poraba posameznih sklopov procesorja Intel Core i5 3570K:
CPE sklop: do cca 50W TDP
iGPE sklop: do cca 27W TDP

7.3 Testna programska oprema in rezultati testiranja

7.3.1 Problem potujočega trgovskega potnika

Problem potujočega trgovskega potnika je eden bolj znanih algoritmov, ki se ga lahko rešuje z golo računsko močjo (t.i. bruteforce). Čas procesiranja začne sčasoma naraščati eksponentno. Zanimivo je, da je tudi na sodobnih procesorjih čas procesiranja z algoritmom surove računske moči relativno dolg že pri majhnem številu mest (npr. 10).

Opisati velja, na kakšen način se problem potujočega trgovskega potnika iz popolnoma serijskega algoritma pretvori v paralelnega. Število permutacij se povečuje s številom mest in se ga izračuna po formuli

$$stevilo\ permutacij = n!,$$

kjer je n število vseh mest. Vsaka permutacija predstavlja zaporedje mest. V programski zanki se za vsako permutacijo kliče metoda, ki izračuna dolžino poti. Permutacije so med seboj neodvisne in zato se jih lahko računa več naenkrat. Za izvedbo programa, ki uporablja vsa procesorska jedra splošnonamenskega procesorja v sodobnih verzijah programskega okolja Microsoft .NET Framework od verzije 4.0 naprej, obstaja knjižnjica TPL (Task Parallel Library), ki poenostavi pisanje programske kode. Ta se iz-

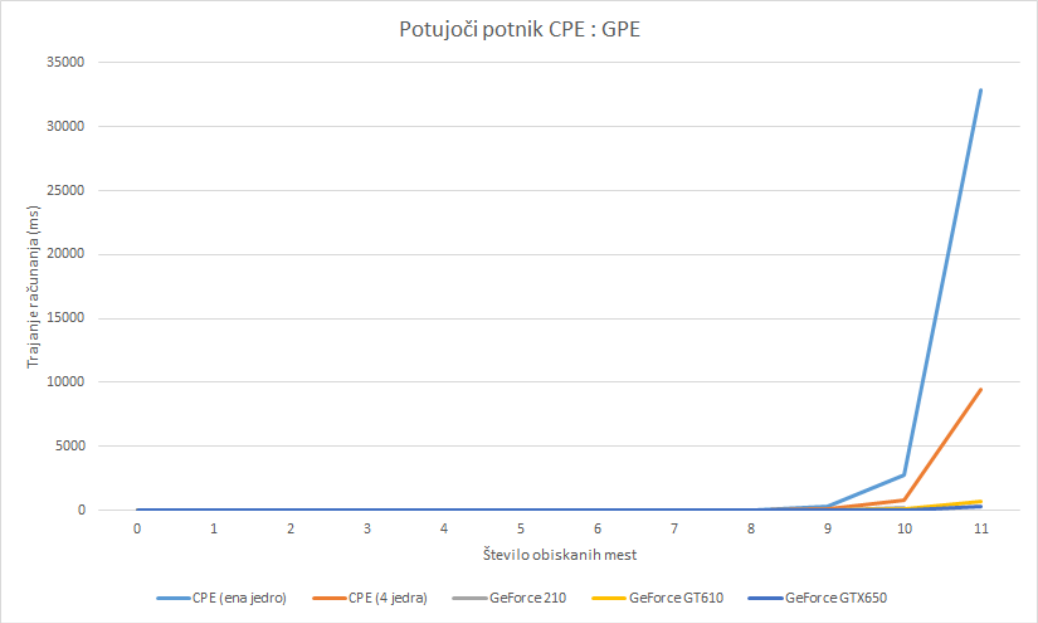
vaja na več procesorskih jedrih splošnonamenskega procesorja. Programerju se tako ni potrebno ukvarjati z nizkonivojskimi mehanizmi, ki dodeljujejo kode procesorskim jedrom na optimalen način, ampak za to nalogo poskrbi samo izvajalno okolje. Paralelna oblika algoritma za več jeder splošnonamenskega procesorja izračuna razdaljo za več permutacij se računa hkrati. Okolje .NET Framework poskrbi, da so vsa procesorska jedra kar se da optimalno zasedena. Pri izvedbi algoritma v paralelni obliki z grafičnim procesorjem je paralelizacija problema podobna, le da tu vsaka programska nit na grafičnem procesorju računa svojo permutacijo. Glede na zasnovo grafičnega procesorja so niti združene v bloke, kjer ima en blok tipično največ 512 ali 1024 niti, odvisno od generacije grafičnega procesorja. Če je glede na število mest permutacij več, kot je lahko programskih niti v enem bloku, potem je problem potrebno razbiti. Tudi število blokov v grafičnem procesorju je strojno omejeno, saj imajo sodobni grafični procesorji tipično lahko $2^{31} - 1$ blokov (starejši grafični procesorji pa 65535). Na grafičnem čipu se torej lahko istočasno izvaja mnogo več programskih niti kot na večjedrnih splošnonamenskih procesorjih, torej se na grafičnem čipu izvaja računanje razdalj za več permutacij kot na večjedrnem CPE.

V grafu 7.1 je razvidno, kako hitro s povečevanjem števila mest permutacij začne naraščati čas procesiranja. Pri algoritmu, ki tako strmo eksponentno narašča pri malo dodanih podatkih, je očitno kako dobro se obnesejo grafični čipi napram splošnonamenskim CPE enotam. Celo grafična kartica GeForce 210, ki je izšla približno 3 leta pred Intel Core i5 3570K ter že ob bila predstavnik najnižjega cenovnega in zmogljivostnega razreda, močno dominira pri računski moči. Omeniti velja, da kljub precej večji porabi CPE procesorja Intel Core i5 3570K (v uporabi vsa 4 računska jedra), po računskih zmogljivostih ni primerljiv grafični kartici GeForce 210.

S tem sem pokazal, da kljub večletni razliki izida na trg tudi najbolj zmogljivi splošnonamenski procesorji (npr. Intel Core procesorji), ne morejo dohajati grafičnih kartic (naj)nižjega cenovnega razreda pri procesiranju paralelnih računskih algoritmov.

V računalniških sistemih, ki so namenjeni paralelnemu procesiranju, so skoraj vedno vgrajene grafične kartice vsaj srednjega zmogljivostnega razreda, ki imajo največjo porabo energije 150W ali več, medtem ko imajo med procesorji tako visoko energijsko porabo samo najbolj zmogljivi procesorji v strežniškem segmentu.

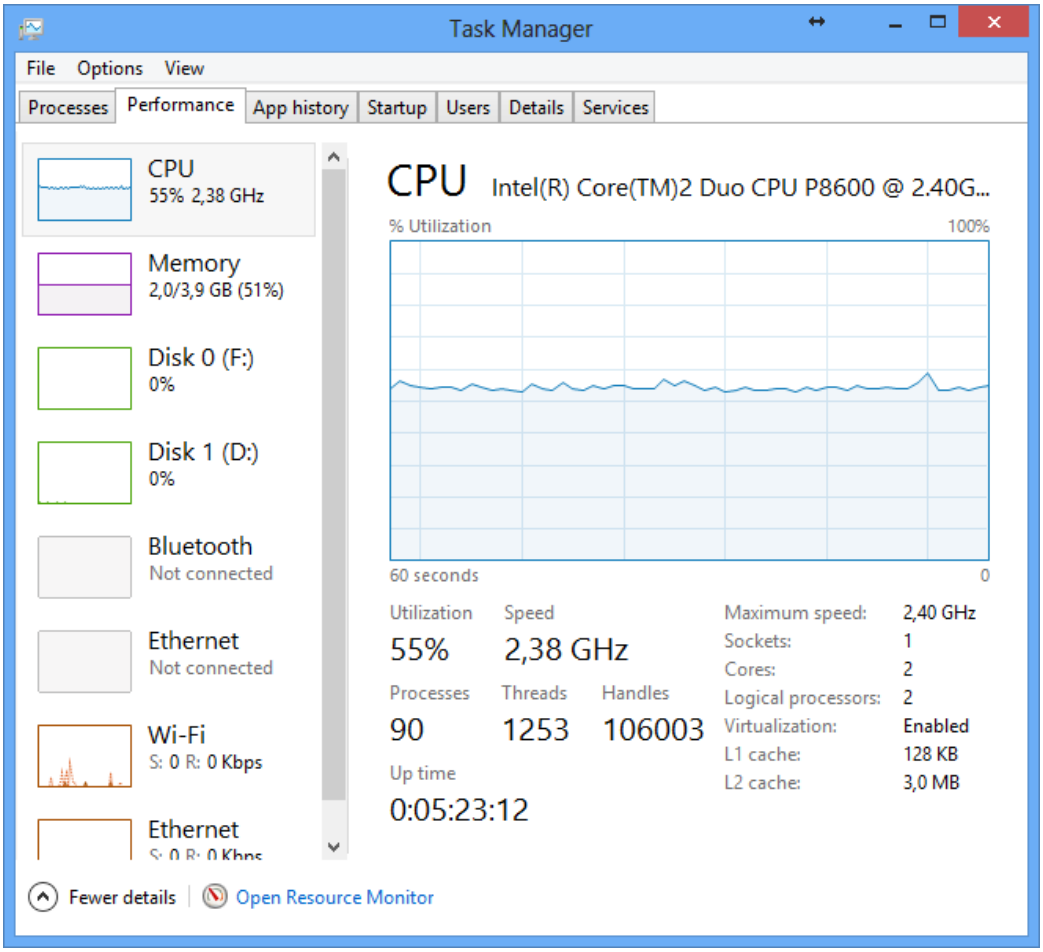
Iz grafa 7.1 je razvidno tudi, da pri problemu trgovskega potnika in temu podobnim paralelnim problemom ena grafična kartica procesira hitreje, kot mnogo sodobnih večjedrnih procesorjev skupaj, četudi ti koristijo vsa procesorska jedra.



Slika 7.1: Naraščanje časa procesiranja pri problemu trgovskega potnika

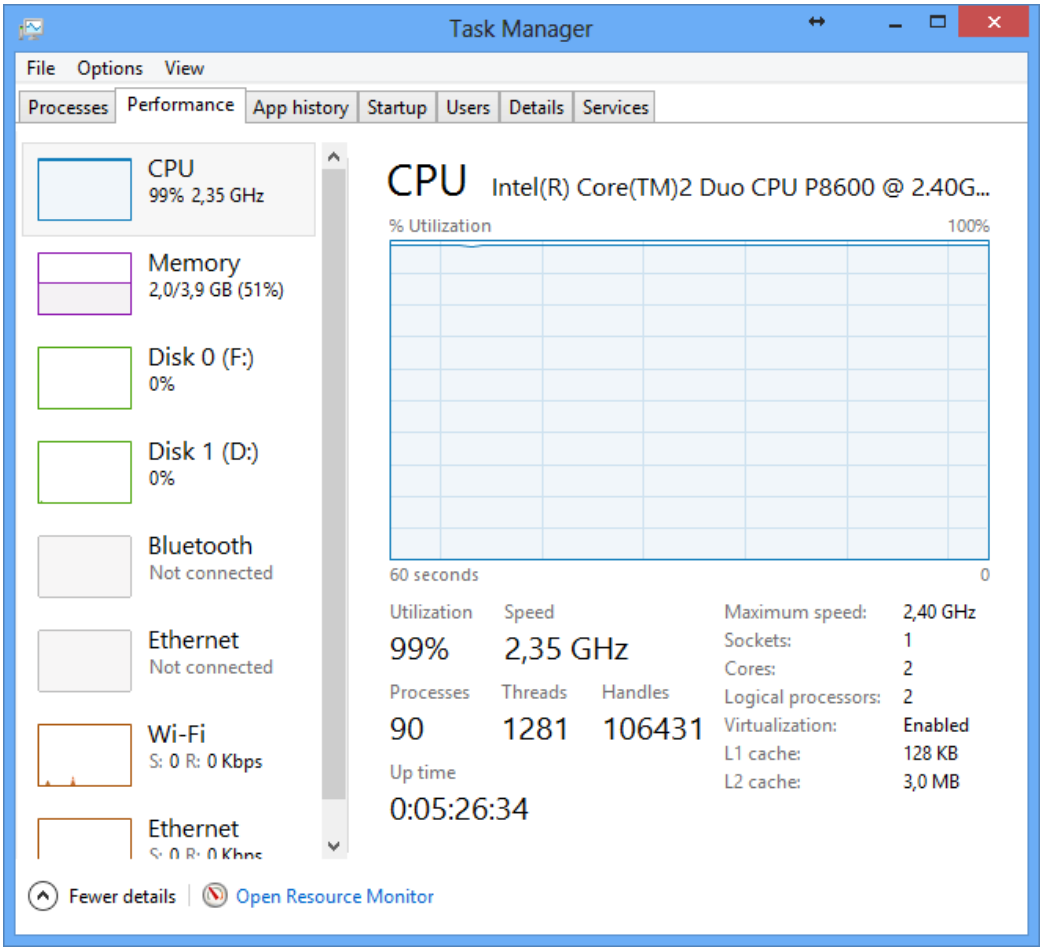
Pri zelo majhnem številu obiskanih mest procesiranje na grafičnih karticah tudi na večih procesorskih jedrih CPE enote traja dlje. Razlog je v kopiranju med glavnim in grafičnim pomnilnikom (pri izvajanju na GPE) ter prevelikih režijskih stroških izvajanja na večih jedrih splošnonamenskega procesorja CPE. Omenjena odstopanja na grafu niso razvidna, saj se to dogaja pri zelo majhnih časih procesiranja in še pri takem računanju je razlika vsega nekaj tisočink sekunde.

Odločil sem se, da algoritem poženem na primerih do vključno 11 mest, ker v primeru CPE enote bi pri dvanajstem mestu že tako narastel čas procesiranja, da bi bila razlika med npr. GeForce GTX650 in GTX680 komaj opazna. Sliki 7.2 in 7.3 prikazujeta obremenjenost procesorskih jeder pri problemu potujočega trgovskega potnika pri večjem številu mest.



Slika 7.2: Izvajanje programa TSP na enem procesorskem jedru (od dveh).

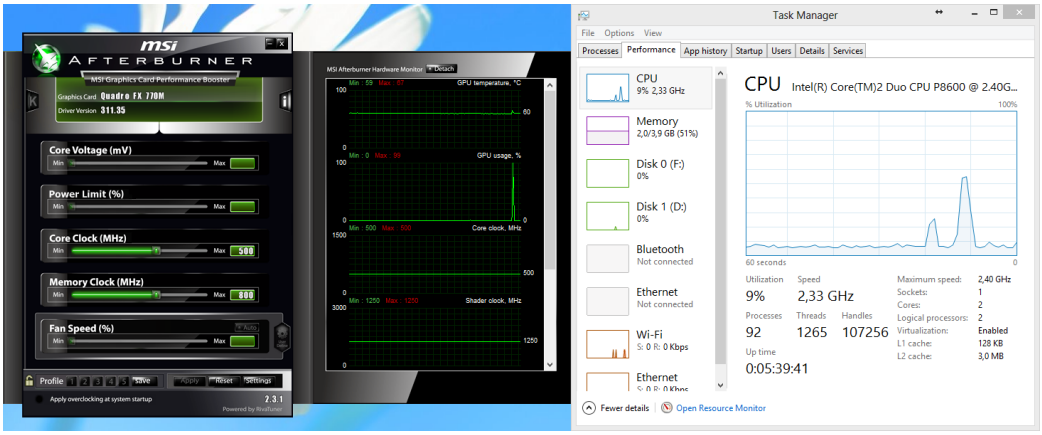
Slika 7.4 prikazuje procesiranje programa TSP na grafični kartici. V programu MSI Afterburner (levo programsko okno) se lepo vidi špica, ki pove, kdaj je grafična sprocesirala dani problem. Za primerjavo je v programu Task Manager (desno) prikazano še stanje osrednjega procesorja v trenutku, ko je problem TSP procesirala grafična kartica. Razvidno je, da sicer zah-



Slika 7.3: Izvajanje programa TSP na vseh procesorskih jedrih (2 jedri).

teve po procesni moči CPE naraščajo, vendar špica ni velika, predvsem pa je časovno gledano za tako velik problem zelo kratkotrajna. Kljub temu, da se algoritem izvaja na grafični, so še vedno prisotni t.i. režijski stroški, ki jih mora sprocesirati osrednji procesor.

Iz povedanega jasno sledi, da četudi se program izvaja na grafični kartici (ali več njih) je hitrost CPE procesorja pomembna, saj se zna zgoditi, da bo grafična kartica delo opravljala hitreje kot bo CPE opravljal režijske stroške. Slednje je mogoče precej zmanjšati, če se enote dela naredi primerno velike za paralelno procesiranje in se jih piše kot t.i. ščepce, ki se izvajajo na grafičnem čipu).



Slika 7.4: Prikaz obremenjenosti grafičnega in osrednjega procesorja med procesiranjem r.cuda.los.

7.3.2 Računanje vidnega polja v neki točki razgleda LOS (Line-of-Sight, GRASS GIS)

Pri računanju vidnega polja v neki točki razgleda sem se pri procesiranju na grafičnih karticah osredotočil še na razmerja med »režijskimi« stroški (kopiranja med glavnim in grafičnim pomnilnikom, dostava podatkov iz hrambe) in časom samega procesiranja na grafični kartici (grafičnem čipu GPE).

V geografskem informacijskem sistemu GRASS GIS se nahaja modul Line-of-Sight, ki računa vidljivost pri neki vhodni GPS točki, višini nad tlemi ter parametru, ki ponazarja največjo zračno razdaljo od izbrane točke. To pomeni, da če je vidljivost iz neke izbrane točke (in neke višine nad tlemi) večja kot jo omejimo s parametrom največje razdalje, se bo preprosto gledalo samo do določene največje razdalje v vidnem polju.

Algoritem programskega modula r.cuda.los deluje tako, da se začne izvajanje računanja pri točki interesa (POI) in se nato pomika proti željeni točki. Področje računanja se razdeli na majhne dele (imenovane rezine). Za vsako rezino na zemljevidu se računa razdaljo in kot pod katerim opazovalec na točki interesa vidi to rezino. Za vsako rezino prevzame izračun svoja programska nit. Računanje omenjenih podatkov, ki je za vsako rezino neodvisno, se zato lahko izvaja vzporedno. Zaradi omenjenega dejstva tudi organizira-

nost niti po blokih nima pravega pomena. Modul r.cuda.los izkorišča globalni pomnilnik katerega slabost je nekaj počasnejši dostop v primerjavi z deljenim (shared) pomnilnikom, vendar to ne igra neke bistvene vloge, saj večji delež časa predstavlja branje in zapisovanje zemljevida na disk. Uporaba globalnega pomnilnika je praktično nujna zaradi velikih zemljevidov. Vhodni podatki v modul r.cuda.los so enaki za vse rezine in jih kot take ni potrebno preračunavati (se shranijo kot konstante). V programski zanki se s pomikanjem po koordinatah x in y (izračune opravljajo niti z indeksi x in y, kar je pomensko precej analogno pomikanju po zemljevidu) približuje željeni končni točki in opravlja že omenjene izračune, ki predstavljajo vidnost točke.[25]

Če si izberemo tako točko, kjer je v resničnem svetu odličen (dolg razgled), lahko z omejevanjem največje razdalje razgleda opazujemo naraščanje časa procesiranja. Omenjeno lastnost lahko dobro izkoristimo za primerjavo hitrosti procesiranja med CPE in GPE procesorji.

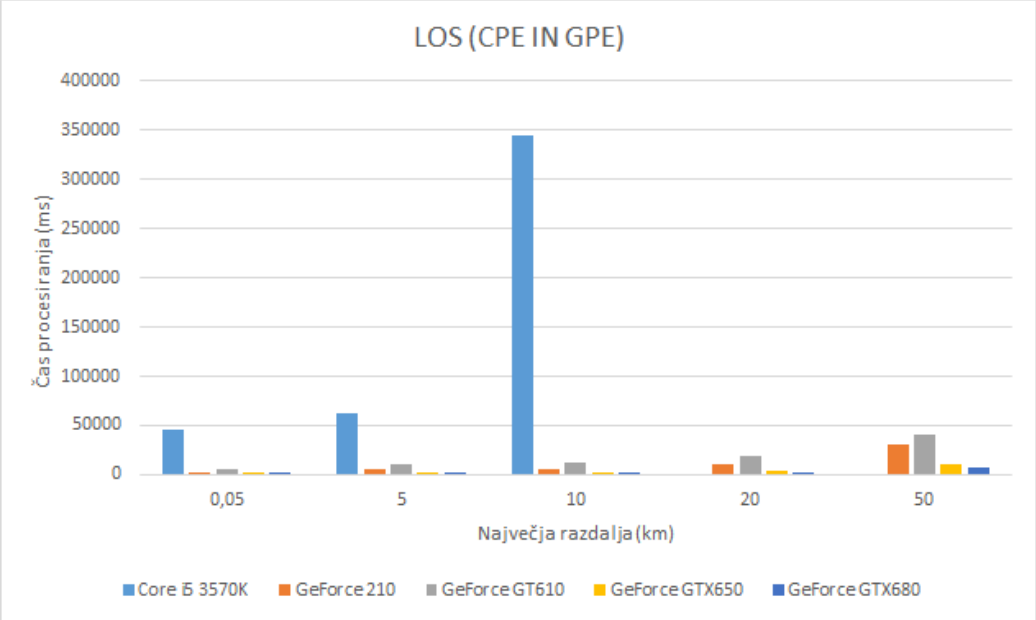
Modul LOS v sistemu GRASS GIS ne glede na število jeder v splošnonamenskem procesorju CPE vedno koristi samo eno, kar sem opazil na monitorju porabe sistemskih sredstev.

Računanje vidnega polja se uporablja pri postavljanju raznih telekomunikacijskih oddajnikov in anten, da se s kar najmanj oddajniki/antenami s signalom najbolje ter najmočneje pokrije neko zemeljsko področje. Za procesiranje vidnega polja obstaja tudi aplikacija napisana v CUDA okolju (avtor aplikacije je Andrej Osterman), ki sicer ni modul programske opreme GRASS GIS, vendar zna brati podatke v obliki kot jih ima GRASS GIS. Tudi vhodni parametri CUDA izvedbe modula LOS so praktično identični tistim v sistemu GRASS GIS.

Na sliki 7.5 so primerjave časov celotnega procesiranja na različnih procesorjih. Omeniti velja, da je procesiranje CPE (Intel Core i5 3570K) enote omejeno na samo eno procesorsko jedro (od štirih), ker LOS modul ni spisan za večjedrne CPE enote.

Testiral sem na vidnih poljih omejenih s 50m, 5000m, 10000m, 20000m in 50000m. Od vključno največje razdelje 10000m naprej sem CPE Intel Core

i5 3570K izključil iz testiranja, saj bi bili časi procesiranja tako veliki, da se rezultatov časov procesiranja na grafičnih karticah nebi več opazilo.



Slika 7.5: Primerjava procesiranja modula LOS na CPU in GPE

Iz pričujočega grafa časov procesiranja na sliki 7.5 je razvidno, da tudi grafične kartice, ki so starejše in so (naj)nižjega zmogljivostnega razreda ter posledično potrebujejo za delovanje najmanj energije, dominirajo nad sodobnim zmogljivim CPE procesorjem.

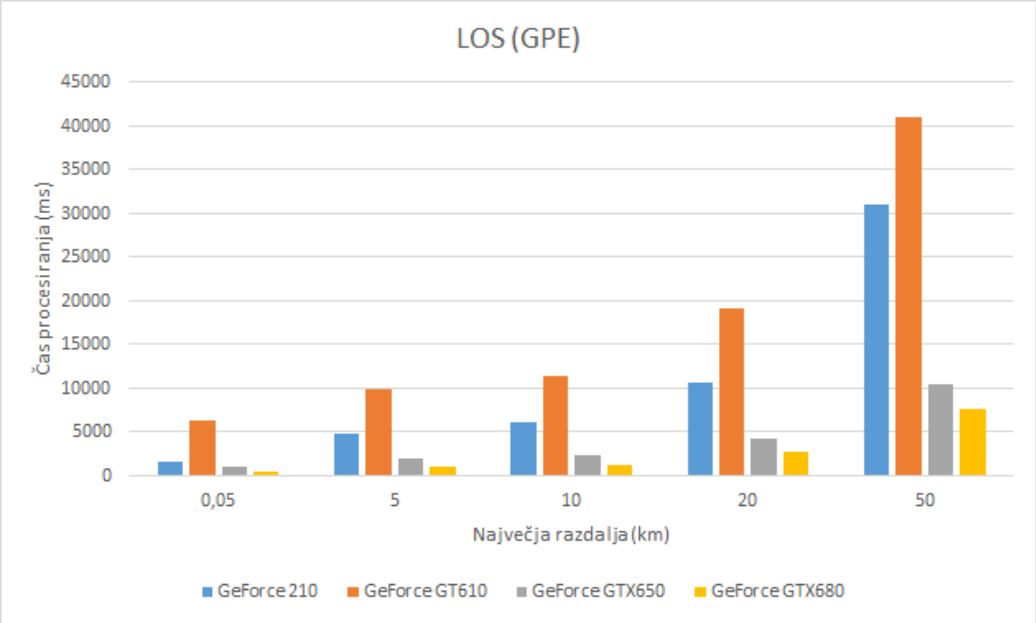
Četudi bi modul LOS v sistemu GRASS GIS koristil vsa 4 jedra, bi v teoriji bil čas procesiranja največ 4-krat manjši (v praksi pa skoraj sigurno ne bi prišlo do 4-kratnega pospeška). V takem primeru bi CPE Intel Core i5 porabljal cca 45W energije (naj še enkrat opomnim, da upoštevam le energijo, ki jo daje splošnonamenski sklop CPE procesorja), kar je malenkost manj kot grafična kartica GeForce GTX650 in več kot kartici GeForce GT610 in GeForce 210 (obe največ 30W energije), ki je kar 3 leta starejša kot Core i5 3570K.

Med testiranjem se mi je pojavila zanimiva anomalija, saj je v CUDA izvedenki LOS modula sistema GRASS GIS precej starejša GeForce 210 kon-

stantno procesirala hitreje kot model GeForce GT610, kljub temu da sta obe kartici najnižjega zmogljivostnega razreda. Kasneje sem ugotovil, zakaj je temu tako. CUDA izvedenko LOS modula sem prevedel za vse verzije t.i. »compute« arhitekture gafičnih čipov. Kartica GeForce 210 ima najstarejšo arhitekturo (najnižjo verzijo) in ne podpira dvojne natančnosti pri številih, kar se je v višjih (novejših) verzijah arhitekture dodalo.

CUDA izvedenka LOS modula pa operira s števili tipa *double*, torej s plavajočo vejico z dvojno natančnostjo. Prevajalnik CUDA sicer prevede tak program za arhitekturo compute 1.1, ki jo ima grafična kartica GeForce 210, vendar pri prevajanju javi opozorilo. Vse operacije z dvojno natančnostjo potem prevajalnik pretvori v enojno natančnost, tako da se aplikacija brez težav izvaja na starejši arhitekturi grafičnega čipa. Tu pa se potem pojavi razlika med GeForce 210 in Geforce GT610. Količina vhodno/izhodnih operacij pri računanju z dvojno natančnostjo v primerjavi z enojno precej naraste (večja kot je surova računska razlika med omenjenima karticama). Močnejši izvedenki, kot sta npr. modela Geforce GTX650 in GeForce GTX680, sta hitrost računanja napram modelu GeForce 210 kompenzirala v precej večjem številu CUDA jeder v samem čipu in tudi v precej višjih urinih frekvencah.

Hitrost samega procesiranja (samo čas izvajanja na t.i. ščepcih) na grafičnih karticah je prikazano na grafu 7.6.



Slika 7.6: Časi izvajanja modula r.cuda.los na grafičnih karticah

Zanimivo je primerjati, kolikokrat se je povečal čas procesiranja kernel-ov (jeder) med največjo razdaljo 50m in 50000m.

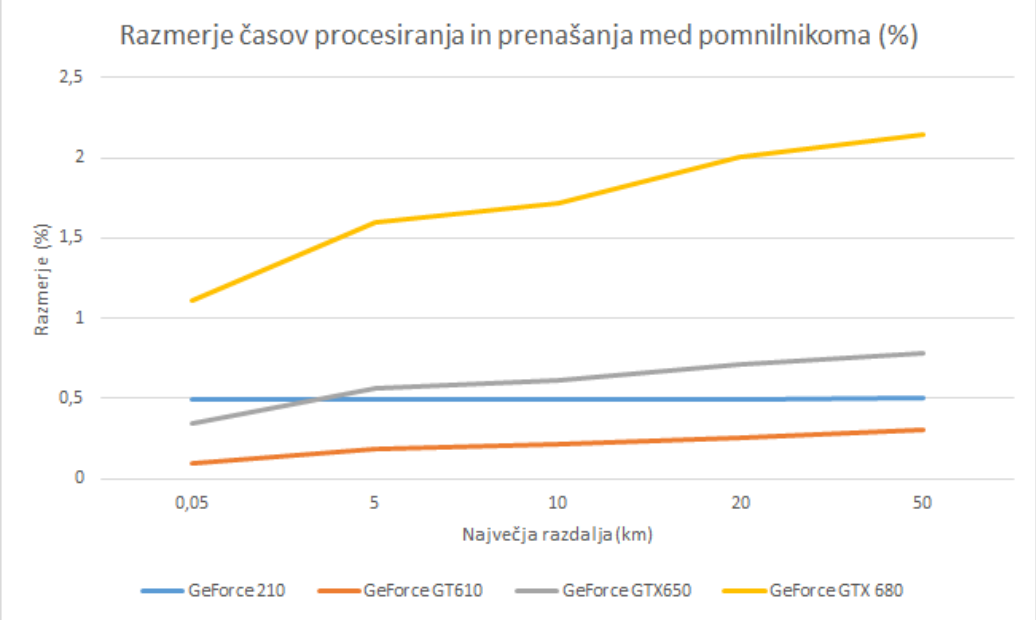
Grafična kartica	**Povečanje časa procesiranja**
GeForce 210	19.3x
GeForce GT610	6.3x
Geforce GTX650	8.9x
GeForce GTX680	10.4x

GeForce GTX650 in GTX680 imata oba RAM tipa GDDR5 (GTX650 5GHz efektivno in GTX680 6GHz efektivno), kar pomeni, da je RAM na modelu GTX680 približno 20% hitrejši kot na modelu GTX650, toda GTX680 ima natanko 4-krat več CUDA jeder kot model GTX650, pri obeh pa je frekvenca samih jeder praktično enaka. To pomeni, da je vpliv hitrosti RAM-a pri modelu GTX680 bistveno večji kot pri modelu GTX650.

Pri manj zmogljivih karticah kot sta GeForce 210 in GT610 je število CUDA jeder in frekvence le-teh tako majhno, da z vidika hitrosti zadostujejo

celo starejši tipi RAM-a. Naslednji zanimiv podatek je razmerje med časom izvajanja (procesiranja) na kernel-ih in med časom kopiranja iz sistemskega pomnilnika računalnika v grafični in nazaj po končanem procesiranju. Pri vseh modelih grafičnih kartic sem opazil, da je čas prenašanja podatkov v obe smeri, torej iz sistemskega pomnilnika v grafični pomnilnik in obratno, enak. Graf 7.7 prikazuje odstotek časa prenašanja med pomnilnikoma (seštevek obeh prenosov) glede na čas procesiranja ščepcev. Z večanjem največje razdalje vidnega polja namreč narašča tudi količina podatkov za računanje.

Opomba k interpretaciji grafa: Seštevek časov prenosa med grafičnim in glavnim pomnilnikom v obe smeri in čas procesiranja znaša 100%. Torej, če je na grafu pri neki razdalji vrednost 2.5% pomeni, da se je 2.5% časa porabilo za prenose med pomnilnikoma (cca 1,25% v vsako smer), 97.5% pa za procesiranje ščepcev na grafični kartici.



Slika 7.7: Razmerje med časom procesiranjanja na GPE in prenašanjem vsebine med glavnim pomnilnikom in pomnilnikom na grafični kartici

Poudariti velja, da prenosi med sistemskim in grafičnim ramom načeloma predstavljajo majhen delež časa v primerjavi s časom procesiranja, vendar

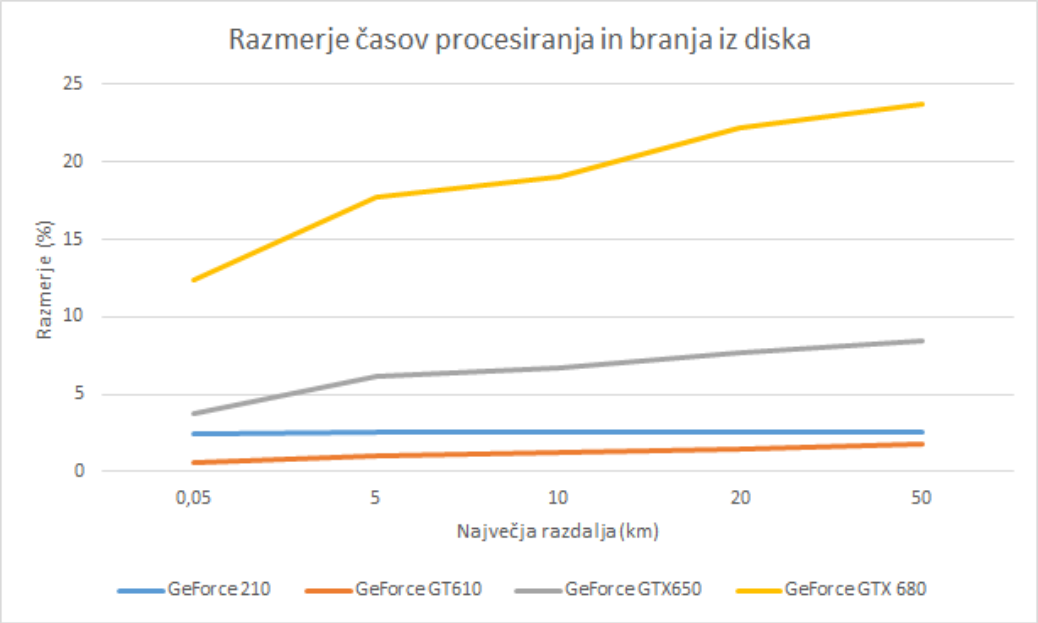
je vseeno mogoče opaziti naraščanje, kar pomeni, da delež časa prenosov med pomnilnikoma narašča. Ta ugotovitev ne preseneča, saj se hitrosti tako grafičnih kot sistemskih pomnilnikov povečujejo precej počasneje, kot narašča število procesorjev znotraj grafičnih čipov. Prav tako same frekvence, kljub močnemu povečevanju procesorskih jeder, naraščajo približno enako hitro kot frekvence pomnilnikov.

Iz grafa je razvidno, da že vstopni modeli srednjega cenovnega razreda grafičnih kartic, ki imajo pomnilnik samo približno 20% počasnejši od najhitrejših izvedenk, včasih potrebujejo hitrejši pomnilnik. Model GeForce GTX680 s samo 20% hitrejšim opmnilnikom od modela GTX650 in 4-krat večjim številom CUDA jeder samo potrjuje hipotezo o (pre)počasnem povečevanju hitrosti pomnilnika.

Zadnja je primerjava deležev dostavljanja podatkov iz strojne opreme, ki trajno hrani podatke. Kot je že zapisano, se je podatke hranilo na SSD-ju Samsung 840 Basic, ki je bil na matično ploščo priklopljen preko vodila SATA 6Gbps (dostikrat poimenovan tudi SATA III).

Na grafu 7.8 je prikazan delež časa branja podatkov iz SSD-ja v glavni sistemski pomnilnik računalnika. Kot hitrost prenosa po vodilu SATA 6Gbps sem vzel 550MB/s, ki je največja praktično dosežena hitrost po omenjenem vodilu (merjeno z več sintetičnimi programi za testiranje hitrosti diskov). Treba je omeniti, da so v praksi hitrosti branja in pisanja lahko precej nižje, predvsem pri slednjem, vendar sem se vseeno odločil, da ilustriram najboljše sceniraije v prid hrambam podatkov in vodilom, ki prenašajo podatke do sistemskega pomnilnika. Iz grafa je kljub temu razvidno, da deleži branja iz SSD-ja predstavljajo precej višji odstotek časa kot kopiranje istih podatkov med pomnilnikoma.

Opomba k interpretaciji grafa: Čas procesiranja ščepcev in branja podatkov iz SSD-ja predstavlja 100%. Če je pri neki razdalji vrednost razmerja 15% pomeni, da se je 15% vsega časa porabilo za branje podatkov iz SSD-ja, ostalih 85% pa za procesiranje ščepcev in prenašanje med pomilnikoma.



Slika 7.8: Razmerje med časom procesiranja ščepcev na GPE in branja datoteke iz diska SSD.

Iz grafa 7.8 je razvidno, da že pri grafični kartici nižjega srednjega zmogljivostnega razreda delež časa branja podatkov iz SSD-ja narašča s količino podatkov. Delež (oz odstotek) časa tu že predstavlja pomemben del k celoti procesiranja. V realnosti so ti deleži lahko še bistveno večji, saj sem vzel najhitrejše možno dostavljanje podatkov iz SSD-ja preko vodila SATA 6Gbps. Tudi za strojno opremo, ki trajno hrani podatke, velja, da se hitrost dostopa in same prenosne hitrosti povečuje precej počasneje kot se povečuje zmogljivost procesorjev, kljub temu, da se je ta v zadnjih letih močno povišala zaradi prehajanja iz klasičnih magnetnih trdih diskov na enote SSD.

Branje podatkov iz datotečnega sistema na SSD-ju si lahko interpretiramo tudi kot poizvedbo na podatkovno bazo (lahko relacijsko ali kakšno drugo tipa NoSQL). Branje podatkov iz podatkovne baze je v večjih in produkcijskih sistemih dandanes praktično nujno in edino pravilno. V takem primeru bi k deležu časa branja podatkov lahko prišteli še izvajanje poizvedbe nad podatkovno bazo ter po možnosti prenos preko računalniškega

omrežja, ki je lahko precej počasno. V takem primeru bi se delež dostave podatkov še bolj povečal. V primeru procesiranja polja vidljivosti sem zaradi preprostosti bral iz navadnega datotečnega sistema in ne iz kake podatkovne baze na oddaljenem računalniku. Na tak način sem tudi dobil precej bolj zanesljive podatke, saj bi v nasprotnem primeru moral upoštevati še stanje omrežja in podatkovne baze, kar bi lahko precej zmanjšalo natančnost merjenja, saj nekaterih dogodkov ni mogoče predvideti. Poenostavitev kljub temu ne zmanjša pomena prenosov podatkov iz podatkovnih shramb.

7.4 Izboljšave hitrosti procesiranja podatkov

Izboljšave, ki omogočijo hitrejšege procesiranje in zato boljšo izkoriščenosti strojne opreme, so možne na več področjih.

7.4.1 Povečevanje števila procesorjev in hitrosti

Število procesorjev v grafičnih karticah se bo nedvomno povečevalo še naprej, saj to zagotavljata oba velika proizvajalca grafičnih čipov (AMD in Nvidia). Tudi frekvence samih procesorjev počasi rastejo, kar še dodatno pripomore k hitrejšemu procesiranju.

Večji del časa se porabi za druge operacije (prenosi med pomnilnikoma in predvsem iz diska v glavni pomnilnik računalniškega sistema), kar pomeni da višanje hitrosti samega procesiranja ni zadosti in si lahko to dejstvo razlagamo s pomočjo Amdahl-ovega zakona.

7.4.2 Hitrejši prenosi med glavnim pomnilnikom in pomnilnikom grafične kartice

Kot je razvidno iz grafov v prejšnjih poglavjih, se za kopiranje večje količine podatkov iz glavnega pomnilnika v pomnilnik grafične kartice (pred procesiranjem) in nato nazaj v glavni pomnilnik (po končanem procesiranju)porabi več časa. Hitrosti glavnega pomnilnika in pomnilnika na grafičnih karticah

se sicer povečujejo, vendar še zdaleč ne tako hitro kakor zmogljivost grafičnih čipov. V podjetju Nvidia so razvoj usmerili v to, da bi lahko grafični čip (kartica) direktno dostopal do glavnega pomnilnika računalniškega sistema, ne da bi bilo potrebno kopirati podatkov med pomnilnikoma. Postavi se vprašanje, če ne bo potem ozko grlo kar osrednji pomnilnik, saj ga lahko sodobni grafični čip naslavlja še hitreje kot osrednji procesor. Omenjeni način bi najbrž prišel prav pri procesiranju manjših količin podatkov, pri katerem bi se s kopiranjem med pomnilnikoma izgubilo več časa, kot bi ga grafični čip porabil zaradi daljšega procesiranja na nekoliko počasnejšem osrednjem pomnilniku.

Kopiranje med pomnilnikoma poteka z maksimalno hitrostjo, saj se prenašajo celi bloki podatkov, ki si v pomnilniku sledijo zaporedno. Osrednji pomnilnik tako lahko koristi t.i. *page mode* način, ki poveča hitrost prenosov tudi do 10-krat. Zaenkrat je to tudi najhitrejši možen način dostopanja do pomnilnika. V prihodnosti ni pričakovati, da bi se asinhronski deli pomnilnikov bistveno pohitrili, kar tudi lahko predstavlja težavo.

Druga možnost so t.i. *skladovni pomnilniki*, ki jih podjetje Nvidia že omenja v prihajajočih generacijah grafičnih kartic. Razlog za nastanek skladovnega pomnilnika je, da se trenutno proizvodni proces povečuje počasneje kot včasih, kar pomeni, da se tudi kapacitete pomnilnikov povečujejo počasneje, zato so proizvajalci pomnilnikov bili primorani najti novo rešitev. Pomnilniški čip se dviguje še v višino in s tem pridobi dodatne plasti, torej več prostora. Napajanje tako grajenega pomnilniškega čipa je po navedbah proizvajalcev precej učinkovitejše, napoveduje pa se tudi dvig hitrosti pomnilnika. Tako se bo lahko istočasno naslavljalo več plasti znotraj enega čipa pomnilnika.

Pri poganjanju programov z različnimi parametri sem opazil, da se s prenosom majhne količine podatkov med sistemskim in grafičnim pomnilnikom prenaša, manjša tudi hitrost prenosa podatkov. Vedeti je potrebno, da ima funkcija *cudamemcpy()* v ozadju režijske stroške alokacije prostora ipd., zato je kljub kopiranju majhne količine podatkov, čas izvajanja t.i. režijskih stro-

škov velik.

7.4.3 Kompresiranje podatkov pred prenosi med pomnilnikoma

Med glavnim pomnilnikom in pomnilnikom grafične kartice se ponavadi prenaša ogromna količina podatkov. Količina podatkov za prenos se lahko zmanjša, če se pred prenosom podatke kompresira in nato, ko prispejo na cilj, spet odkompresira. Pred procesiranjem je cilj grafični pomnilnik, po procesiranju pa glavni pomnilnik.

Pri tej izboljšavi se pojavita dva večja problema. Splošno kompresiranje, ki naj bi se izvajalo avtomatsko npr. na nivoju strojne opreme ali programskega okolja, ne daje tako dobrih rezultatov kot namensko. Dober primer so naprimer video datoteke, ki jih kompresira nek enkoder in jih predvajalniki zmorejo predvajati tudi v kompresirani obliki, in pa splošno kompresiranje, ki ga ponujajo aplikacije kot so WinZIP, 7-Zip, WinRAR,.... Drugi večji problem so sami algoritmi za kompresijo podatkov, ki so povečini serijske (zaporedne) narave. Kljub številnim poskusom algoritma, ki bi v osnovi dajal dobre rezultate in bi bil paralelen, (še) ni. Obstajajo le paralelne izvedenke serijskih algoritmov, ki sicer precej dvignejo hitrost samega kompresiranja podatkov, vendar na račun nižje kompresije. Stopnja kompresije, ki jo vzporedno izvaja več procesorjev, je manjša od stopnje kompresije, ki jo izvaja eno samo procesorsko jedro. Tega ni težko razumeti, saj se morajo podatki, ki bi jih radi kompresirali, razdeliti v manjše dele, nato vsak procesor nad njimi izvaja kompresijo. Več kot je teh procesorjev, manjše so enote kompresiranja, pri manjši enoti pa je tudi učinkovitost stiskanja manjša. V primeru, ko stiskanje opravlja eno samo procesorsko jedro, se kompresija vrši nad vsemi podatki.

Stiskanje podatkov pred kopiranjem na drug pomnilnik mora biti hitra operacija, saj se v nasprotnem primeru porabi več časa, kot če bi prenašali kar nestisnjene podatke. V kolikor programer meni, da bi s stiskanjem podatkov skrajšal čas prenosa med pomnilnikoma, mora tudi sam poskrbeti za to.

Možnost je, da grafika izvaja procesiranje nad stisnjenimi podatki, vendar mora za to poskrbeti programer, je pa ta pristop v praksi redko uporabljan, saj je zahteven za implementacijo in največkrat neuporaben za nadaljno uporabo v drugih aplikacijah.

7.4.4 Hitrost zajema podatkov namenjenih za procesiranje

Procesiranje pa ne zajema samo dogajanja, ko so podatki že shranjeni v pomnilniku računalniškega sistema. Navadno so podatki shranjeni v podatkovni bazi (relacijski ali NoSQL bazi), pred procesiranjem pa je potrebno narediti poizvedbo in tako dobiti podatke, ki so relavantni za obdelavo na grafičnem procesorju. To je pomembno, kadar računalniški sistem izvaja procesiranje nad t.i. živimi podatki in to predstavlja kontinuiran proces v poslovnem okolju (npr. spremljanje prometa).

Čas procesiranja na grafičnem čipu lahko predstavlja majhen odstotek celotnega časa, če podatki niso dovolj hitro dostavljeni iz strani podatkovne baze. Dodaten problem predstavljajo tudi hitrosti prenosnih poti, če so podatki v drugem računalniškem sistemu. V prihodnjih letih bo zato precej pomembno, da bo podatkovna baza v kar se da velikem deležu v glavnem pomnilniku oz. da bo računalniški sistem operiral s podatkovno bazo v pomnilniku, v ozadju pa bodo spremembe shranjevale na podatkovni medij, ki ob izgubi napajanje ne izgubi še podatkov (trdi diski, flash diski). V sistemih, kjer konstantna sinhronizacija podatkov ni bistvena, se lahko namesto relacijske podatkovne baze uporabi primerno bazo tipa NoSQL in se tako še s procesiranjem poizvedb porabi manj računalniških sredstev, kot bi jih za enako delo porabili pri relacijski bazi.

Poglavje 8

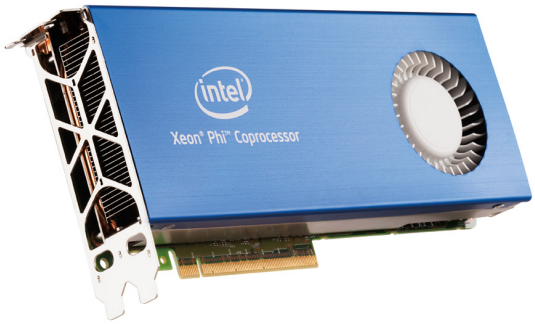
Zaključek

V prihodnosti se bo po pričakovanju razkorak med GPE in CPE še povečal, saj število jeder v osrednjih procesorjih narašča počasi, enako velja za frekvence samih jeder znotraj procesorja, medtem ko je dogajanje na področju razvoja grafičnih čipov ravno nasprotno. Število procesorjev znotraj grafičnega čipa se v letu in pol podvoji, kljub temu pa relativno hitro raste tudi sama frekvenca teh procesorjev. Proizvajalci grafičnih procesorjev napovedujejo hitro povečevanje zmogljivosti grafičnih čipov v bližnji in srednji prihodnosti, kar posledično pomeni večje število jeder. Slednji so že v prototipnih fazah ali fazi razhroščevanja. Na področju razvoja osrednjih procesorjev, tj. procesorjev, ki ponujajo najvišjo zmogljivost in tistih v mobilnih napravah, je slika drugačna. Oba največja proizvajalca CPE enot omenjenega hitrostnega razreda namreč že dlje časa povečujeta obdobje med izdajami novih serij. Prav tako tudi same pohitritve niso več tako izrazite, kot so bile naprimer pred letom 2000.

Prednost pisanja paralelnih algoritmov za CPE procesorje napram grafičnim procesorjem je, da trenutno obstajajo razhroščevalniki (ang.: debugger), ki so veliko bolj dodelani in programerju prijazni. Prav tako je drugačno tudi pisanje kode, saj podatkov ni treba nalagati iz osrednjega pomnilnika v grafični in po končanem procesiranju spet nazaj.

Zaradi omenjenega dejstva je Intel razvil produkt, torej splošnonamenski

procesor, ki ima ogromno jeder in je realiziran v obliki kartične rešitve. Jedra samega procesorja imajo nizko frekvenco, vendar jih je na silicijevi rezini več 10. Najhitrejši CPE procesorji z visoko frekvenco danes tipično nimajo več kot 6 jeder oz. posledično 12 logičnih procesorjev pri Intel-ovih procesorjih zaradi tehnologije Hyper-Threading, ki operacijskemu sistemu računalnika pokaže 2 logični jedri za eno fizično jedro.



Slika 8.1: Intel-ova večjedrna kartična rešitev Xeon Phi. *http://images.anandtech.com/doci/6265/Intel_Xeon_Phi_PCIe_Card.jpg*

Testiranje z omenjenim produktom žal ni bilo mogoče, saj je v času pisanja na tržišču še ni bil pogost. Največji toplotni pečat, ki ga oddaja ta kartica je na nivoju najhitrejših grafičnih kartic (225-300W) in ima 57-61 jeder (odvisno od modela), frekvence 1.1-1.2GHz in cene od $1600 do preko $4000. Omenjeni produkt bi bila zanimivo primerjati z grafičnimi karticami pri implementaciji algoritma trgovskega potnika, saj je arhitektura jeder identična kot pri običajnih osrednjih procesorjih. Primerjava z modulom r.los GIS sistema GRASS bi bila nesmiselna, saj je ta, kot sem že omenil, pisan za eno procesorsko jedro in bi se zaradi počasnosti enega samega jedra Xeon Phi odrezal porazno.

Literatura

[1] Hilbert, Martin; López, Priscila (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". Science 332 (6025): 60–65. doi:10.1126/science.1200970. PMID 21310967

[2] Walter C: Kryder's Law. Sci Am August 2005, 293:32-33

[3] Krewell, Kevin. "Intel cancels 4ghz p4."Microprocessor Report 18.14 (2004): 1-3.

[4] Kim, JunSeong, and Jongsu Yi. "Performance sensitivity of SPEC CPU2000 over operating frequency."Proceedings of the 2004 international symposium on Information and communication technologies. Trinity College Dublin, 2004.

[5] Thomas, C. Douglas, and Alan E. Thomas. "Method and system for controlling a processor's clock frequency in accordance with the processor's temperature."U.S. Patent No. 5,752,011. 12 May 1998.

[6] Megiddo, Nimrod. "Applying parallel computation algorithms in the design of serial algorithms."Journal of the ACM (JACM) 30.4 (1983): 852-865.

[7] Gwennap, Linley. "Intel, HP make EPIC disclosure."Microprocessor report 11.14 (1997): 1-9.

[8] Ramseyer, Richard R., and Andries van Dam. "A multi-microprocessor implementation of a general purpose pipelined CPU."ACM SIGARCH Computer Architecture News. Vol. 5. No. 7. ACM, 1977.

[9] Stone, Harold S., and John Cocke. Čomputer architecture in the 1990s.Čomputer 24.9 (1991): 30-38.

[10] Brodal, Gerth Stølting, and Gabriel Moruz. Škewed binary search trees."Algorithms–ESA 2006. Springer Berlin Heidelberg, 2006. 708-719.

[11] http://www.intel.com/support/processors/mobile/core2duo/sb/CS-023242.htm

[12] Lee, Johnny KF, and Alan Jay Smith. "Branch prediction strategies and branch target buffer design.Čomputer 17.1 (1984): 6-22.

[13] http://www.nvidia.com/docs/IO/121761/adobe-hardware-performance-white-paper-2012-06-27.pdf

[14] Da Costa, Georges, et al. "The green-net framework: Energy efficiency in large scale distributed systems."Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE, 2009.

[15] Kirk, David B., and W. Hwu Wen-mei. Programming massively parallel processors: a hands-on approach. Newnes, 2012.

[16] Ujaldon, Manuel. "High performance computing and simulations on the GPU using CUDA."High Performance Computing and Simulation (HPCS), 2012 International Conference on. IEEE, 2012.

[17] Bonaccorsi, Enrico, et al. "Infiniband Event-Builder Architecture Test-beds for Full Rate Data Acquisition in LHCb."Journal of Physics: Conference Series. Vol. 331. No. 2. IOP Publishing, 2011.

[18] http://www.amd.com/us/products/desktop/graphics/7000/7990/pages/radeon-7990.aspx

[19] Kay, Rony. "Pragmatic Network Latency Engineering Fundamental Facts and Analysis.ČPacket Networks, URL: http://www. cpacket. com (as of 2012-02-06) (2009).

[20] Liu, Jiuxing, et al. "Performance comparison of MPI implementations over InfiniBand, Myrinet and Quadrics.Šupercomputing, 2003 ACM/IEEE Conference. IEEE, 2003.

[21] http://www.memorybenchmark.net/ram.php?ram=Corsair+CM2X1024-6400C4+1GB&id=341

[22] Molka, Daniel, et al. "Memory performance and cache coherency effects on an Intel Nehalem multiprocessor system."Parallel Architectures and Compilation Techniques, 2009. PACT'09. 18th International Conference on. IEEE, 2009.

[23] https://www.change.org/petitions/advanced-micro-devices-fix-bugs-in-opencl-compiler-drivers-and-eventually-opensource-them

[24] Tudorica, Bodgan George, and Cristian Bucur. "A comparison between several NoSQL databases with comments and notes."Roedunet International Conference (RoEduNet), 2011 10th. IEEE, 2011.

[25] Andrej Osterman. "Implementacija modula r.cuda.los v odprtokodnem paketu GRASS GIS z vzporednim računanjem na gra?čnih karticah NVIDIA CUDA."ELEKTROTEHNISKI VESTNIK 79(1-2): 19–24, 2012.